JACQUES BOISGONTIER ET SOPHIE BREBION



ISBN : 2 86595.098.0

ORIC-1
pour tous

INITIATION GENERALE

Visa pour l'informatique — Jean-Michel Jégo
Mon ordinateur — Jean-Claude Barbance
L'ordinateur individuel — Yves Leclerc

INITIATION AUX MATERIELS :

Collection « ... POUR TOUS »

ORIC-1 pour tous — Jacques Boisgontier et Sophie Brébion
52 programmes, ORIC-1 pour tous — Jacques Boisgontier

COLLECTION « MATERIELS » — Série verte

La découverte de l'Apple II — tome 1 — par Frédéric Lévy et Dominique Schraen
La découverte du CBM — par Daniel-Jean David
La découverte du VIC — par Daniel-Jean David
La découverte du PC-1211 — par Jean-Pierre Richard
La découverte de la TI 57 — par Xavier de la Tullaye
La découverte du PC-1251 — par Jean-Pierre Richard
La découverte du PC-1500 — par Jean-Pierre Richard
La découverte du FX 702P — par Jean-Pierre Richard
La découverte du TI-99/4A — par Frédéric Lévy et Dominique Schraen
La découverte de l'Atari — par Daniel-Jean David
La découverte du PB-100 — par Pierrick Moigneau
La découverte du Goupil — par Jean-Yves Michel
La découverte du TO 7 — par Dominique Schraen et Maurice Charbit

INITIATION AUX LANGAGES

Collection « ... POUR TOUS »

BASIC pour tous — Jacques Boisgontier et Sophie Brébion

Collection « LANGAGES » série verte

Langages de programmation — Stéphane Berche et Claude Lhermitte
Programmer en Basic — Michel Plouin
Programmer en LSE — Stéphane Berche et Yves Noyelle

ISBN : 2-86595-098-0

ORIC-1 pour tous

par
Jacques Boisgontier
et
Sophie Brébion



EDITIONS DU P.S.I.
1984

Jacques Boisgontier a également publié aux éditions du P.S.I. :
— Le Basic et ses fichiers - tomes 1 et 2
— L'Apple et ses fichiers - tome 1
— Le Basic de A à Z
— Basic pour tous

SOMMAIRE

COMMENT SE PRÉSENTE VOTRE ORDINATEUR ?

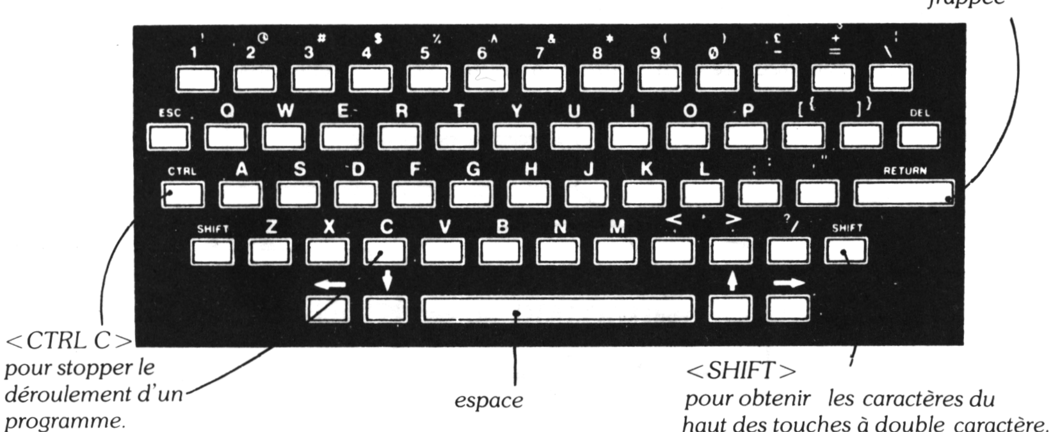
Vous avez décidé de vous initier au langage BASIC, et pour cet apprentissage vous avez choisi l'ORIC-1.
Vous êtes face à votre ordinateur, livre en main.

L'écran :

Il s'agit d'un écran bien familier, celui de votre téléviseur relié à l'ORIC-1 grâce à la prise péritélévision.
C'est par l'intermédiaire de l'écran que l'ordinateur répond à vos questions.
Mais comment questionner l'ordinateur ?
L'ordinateur n'étant pas encore en mesure d'accepter les ordres vocaux, vous devez utiliser un autre intermédiaire, le clavier.

Le clavier :

<RETURN> pour valider une ligne frappée



Comme vous le constatez, il ressemble fort à celui d'une machine à écrire, lettres, chiffres, signes de ponctuation, symboles mathématiques, barre d'espacement, tout est là.

Cependant, par rapport à un clavier classique vous observez certaines différences :
— La première, la plus frappante, c'est la disposition des lettres. L'ORIC-1 est en effet doté d'un clavier QWERTY, comme tous les claviers fabriqués outre-Manche.
— La seconde est un ensemble de touches inhabituelles :
DEL sert à effacer un caractère erroné
← sert à déplacer le curseur vers la gauche de l'écran

→ sert à déplacer le curseur vers la droite de l'écran
↑ sert à déplacer le curseur vers le haut de l'écran
↓ sert à déplacer le curseur vers le bas de l'écran

C'est donc grâce à ce clavier que vous allez pouvoir communiquer, DIALOGUER, avec l'ordinateur.

Pour vous familiariser avec ce nouveau mode de communication, tapez (doucement !) sur toutes les touches sans appréhension. Vous ne risquez rien... l'ordinateur non plus.

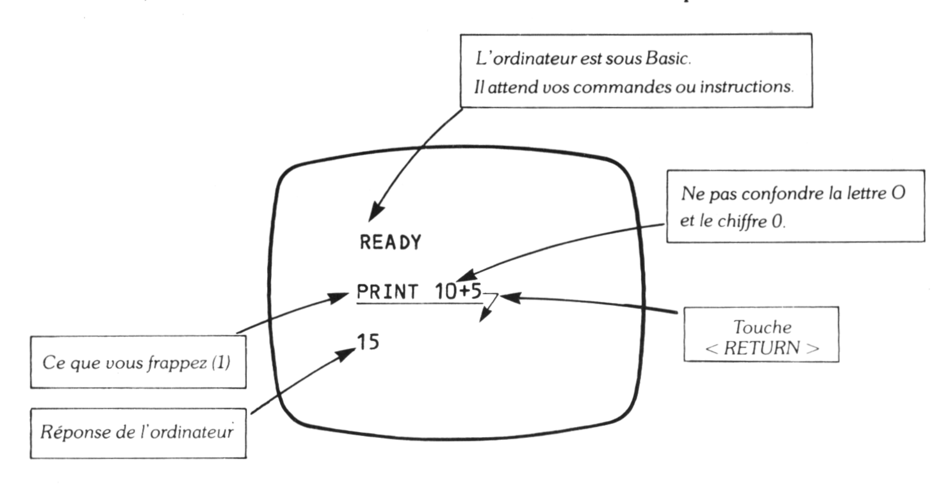
Tandis que vous vous familiarisiez avec le clavier, vous avez découvert certaines touches encore inconnues.

RETURN :

C'est à l'aide de cette touche que vous 'validez' le message que vous voulez envoyer à l'ordinateur. Tant que vous n'avez pas appuyé sur cette touche, vous pouvez effacer les caractères frappés avec la touche DEL.

Appuyer sur la touche <RETURN> pour valider la frappe d'une instruction ou d'une commande doit donc devenir un réflexe.

Si vous l'oubliez, l'ordinateur restera sourd... à toutes vos imprécations !



Remarque : Le message READY qui indique que vous êtes sous BASIC sur TRS-80 devient OK sur d'autres machines.

(1) Nous avons choisi de SOULIGNER FICTIVEMENT sur les représentations d'écran ce que vous frappez pour le différencier visuellement des réponses de l'ordinateur.

DEL :

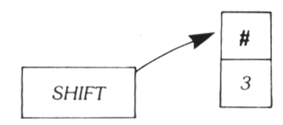
Cette touche vous permet, tant que vous n'avez pas appuyé sur <RETURN>, d'effacer le dernier caractère frappé.
Le curseur se déplace alors de droite à gauche en effaçant le dernier caractère, l'avant-dernier et ainsi de suite au fur et à mesure de sa progression vers la gauche.

Vous avez compris l'utilité de cette touche que vous utiliserez pour corriger vos fautes de frappe, si toutefois vous vous en êtes aperçu AVANT d'avoir validé votre ligne par ?... <RETURN>.
Sinon, patience, nous vous expliquerons un peu plus tard comment modifier une ligne déjà créée (voir annexe).

SHIFT :

Vous utilisez cette touche pour frapper le caractère supérieur des touches à double caractère.

Par exemple :



Pour frapper le caractère #.

ATTENTION, à la différence des machines à écrire, la touche SHIFT ne reste pas enfoncée tandis que vous frappez le caractère désiré. Vous devez d'ABORD appuyer sur la touche SHIFT puis appuyer sur le caractère désiré TOUT EN MAINTENANT LA TOUCHE SHIFT.

En bref

- ***LE CLAVIER*** *vous permet de* ***DIALOGUER*** *avec l'ordinateur.*

- ***L'ECRAN*** *vous donne la possibilité de* ***VISUALISER*** *cette 'conversation' qui s'établit entre vous et la machine.*

- *La touche retour chariot,* **RETURN,** ***VALIDE*** *la frappe de l'information transmise à l'ordinateur.*

- *La touche* **DEL** *déplace le curseur en arrière en effaçant les signes sur son passage.*

- *La touche* **SHIFT** *est utilisée pour frapper le caractère du haut des touches à double caractère.*

LE MODE IMMÉDIAT (Appelé aussi MODE DIRECT)

Pour dialoguer avec l'ordinateur, il ne suffit malheureusement pas de bien connaître son clavier, il faut également parler sa langue, le BASIC. Ne commencez pas à vous inquiéter ; vous verrez comme c'est simple !
Un bref apprentissage vous suffira pour devenir parfaitement bilingue.

Ceci dit, pour initier la conversation, il vous suffit de frapper une instruction que vous n'omettrez pas de valider par <RETURN>. L'ordinateur vous répond alors IMMEDIATEMENT.

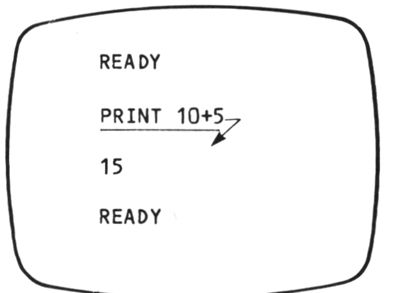
Commençons :

Version Anglaise

— Vous : PRINT 10 + 5
— Lui : 15

Version Française

— Vous : ECRIS 10 + 5
— Lui : 15

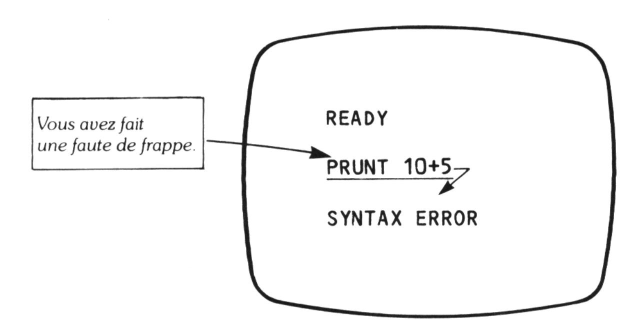


PRET
ECRIS 10+5
15
PRET

Regardons de plus près l'écran version B A S I C

(pour débutants) Beginner's ← B
(tous usages) All-purpose ← A
Symbolic ← S
Instruction ← I
Code ← C

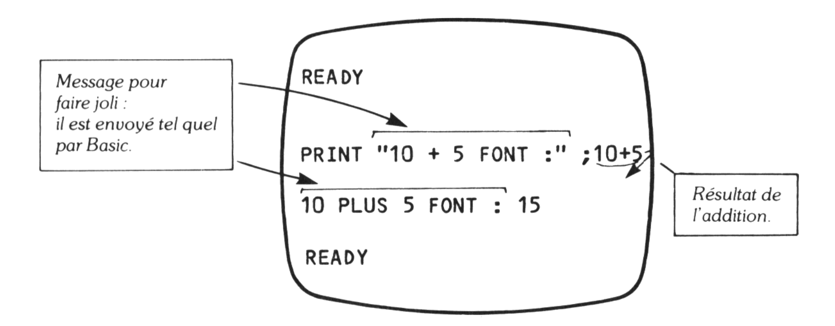
Comme pour toutes les langues, il existe en BASIC une SYNTAXE que vous devez respecter. Dans le cas contraire, l'ordinateur ne comprend pas le message que vous lui transmettez. Il vous le signale alors en affichant le message SYNTAX ERROR (erreur de syntaxe) et attend que vous frappiez de nouveau, sans faute, votre instruction.



Imaginons que vous vous soyez aperçu de cette impardonnable (nous parlons au nom de la machine !) faute de frappe avant d'avoir appuyé sur la touche <RETURN>.
Qu'auriez-vous pu faire pour remédier à cet oubli ?

Comme vous l'aurez remarqué, l'ordinateur se contente de vous donner une réponse sans faire de discours.
C'est à vous que revient le soin d'améliorer la présentation de sa réponse.

Pour ce faire, il vous suffit d'écrire un MESSAGE placé ENTRE GUILLEMETS.

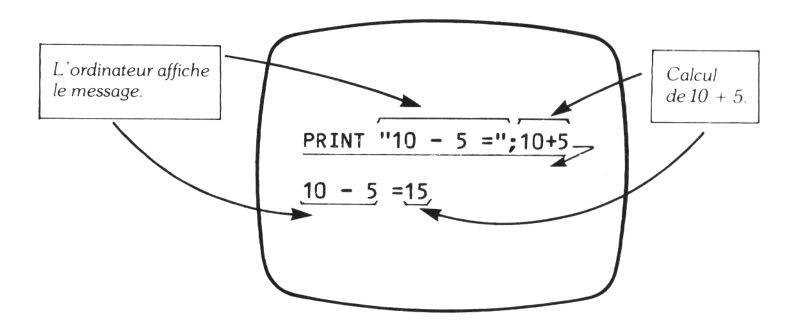


ATTENTION AUX GUILLEMETS !

Ce sont les guillemets qui préviennent l'ordinateur qu'il s'agit là d'un message.
Ce message est parfaitement arbitraire. Vous auriez pu tout aussi bien frapper :

```
PRINT "10 - 5 =";10+5
```

L'ordinateur mémorise le message placé entre guillemets et l'affiche textuellement.



Ceci vous permet de constater sa fidélité, certes ! mais aussi son manque 'd'intelligence'.

En bref

• *Le mode* ***IMMEDIAT*** *(ou* ***DIRECT****) vous permet de connaître 'immédiatement' l'effet d'une instruction.*

• *L'instruction* **PRINT** ***(ECRIS)*** *est employée pour afficher à l'écran.*

• *Le message* ***SYNTAX ERROR*** *indique que l'ordinateur n'a pas compris ce que vous lui demandiez. Impuissant, il vous demande de bien vouloir frapper à nouveau l'instruction sans vous tromper dans la formulation.*

• *Les* ***GUILLEMETS*** *«...» indiquent à la machine qu'elle doit* ***SEULEMENT*** *reproduire le message qu'ils encadrent.*

MÉMORISATION DES VALEURS : VARIABLES

L'instruction PRINT, pour importante qu'elle soit, n'est cependant guère révélatrice des possibilités de la machine.

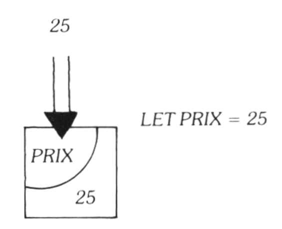
L'instruction LET variable = valeur :

Cette instruction permet de conserver en 'mémoire' des valeurs qui seront ensuite utilisées (pour des calculs par exemple).

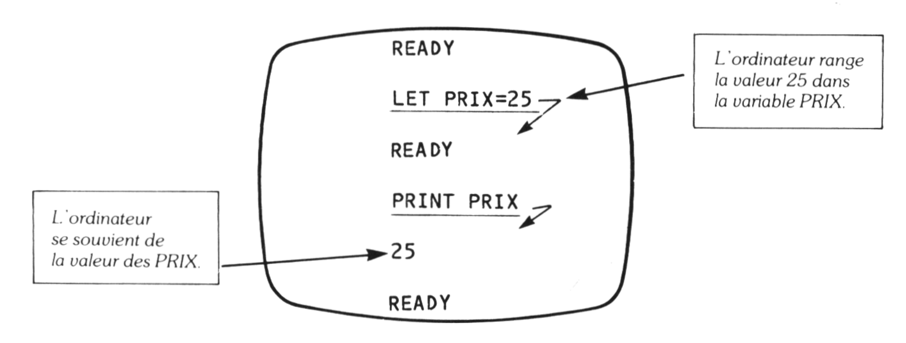
Pour schématiser la notion de variable, comparons-la à une case (c'est d'ailleurs ainsi que nous la symboliserons) à l'intérieur de laquelle nous rangeons une valeur. Puis donnons lui un NOM afin de l'IDENTIFIER.
Ainsi pour AFFECTER (donner) la valeur 25 à une variable que nous appelons 'PRIX', nous écrivons :

```
LET PRIX=25
```

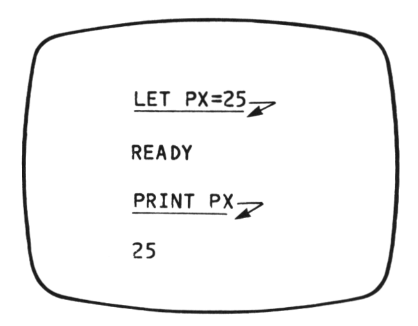
BASIC range alors la valeur 25 dans une case PRIX.



L'ordinateur se souvient de la valeur de PRIX. Pour afficher cette valeur, il suffit de frapper PRINT PRIX.

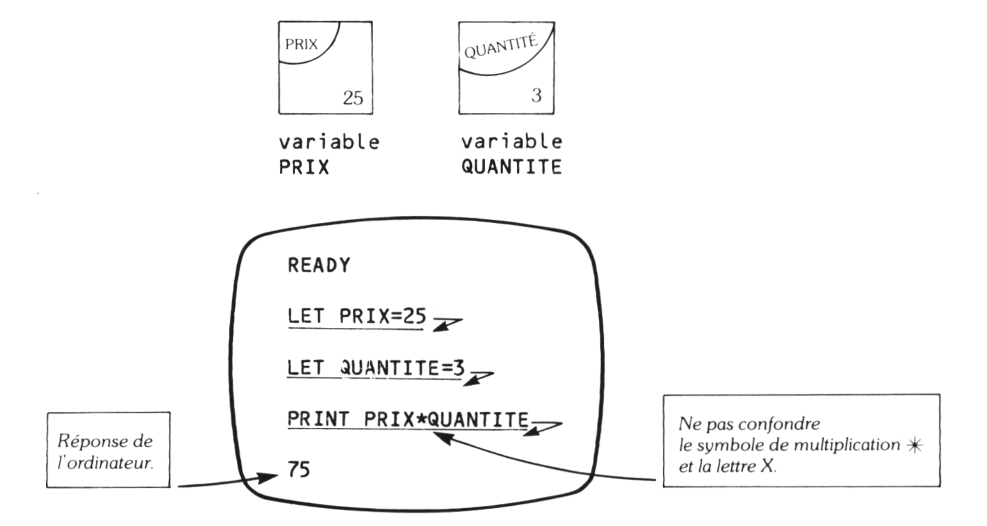


Le nom donné à la variable, ici PRIX, est arbitraire, pour le vérifier, faites :

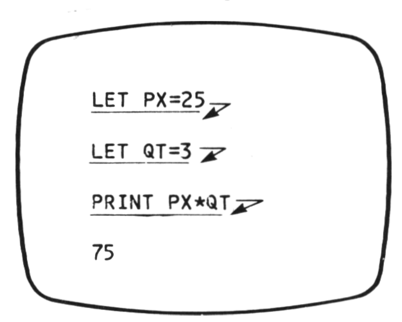


Il est cependant important que le nom variable choisi soit le plus mnémonique possible.

Calculons maintenant le total d'un produit de 25 Francs pour une quantité de 3.
Pour ce faire, nous utilisons 2 variables (2 cases).



Les noms de variable 'PRIX' et 'QUANTITE' pourraient être remplacés par 'PX' et 'QT'



Nous obtiendrions le même résultat en écrivant :

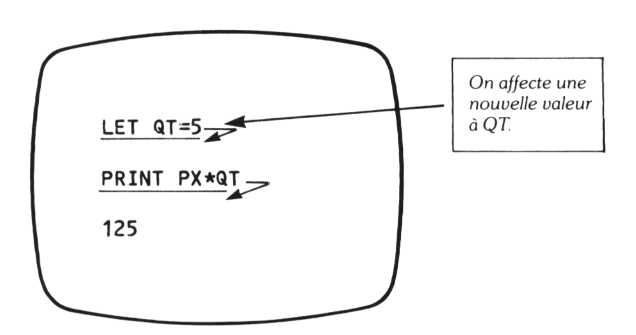
```
LET PX=25
LET QT=3
LET T=PX*QT
PRINT T

75
```

Voyons maintenant comment changer la valeur de la variable QT. Pour lui donner la valeur 5, il suffit de faire :

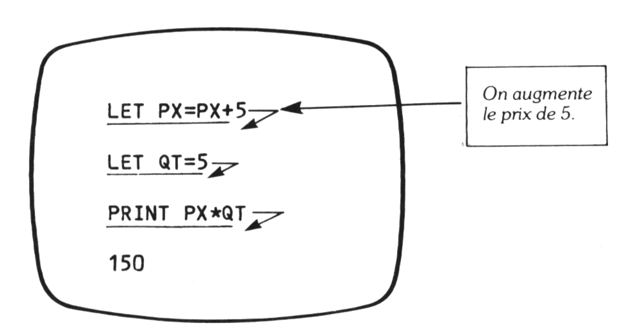
```
LET QT=5
```

(la variable PX, elle, conserve son ancienne valeur)



Naturellement, l'ordinateur 'oublie' l'ancienne valeur (3) de QT.

Imaginons maintenant que le prix augmente de 5 Francs. Nous pourrions bien sûr faire : LET PX = 30
Mais nous pouvons également faire :

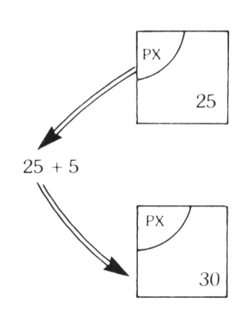


LET PX = PX + 5 ne doit pas être compris comme une égalité algébrique. L'ordinateur interprète cette insertion ainsi :

① **Il prend le contenu de la case PX**

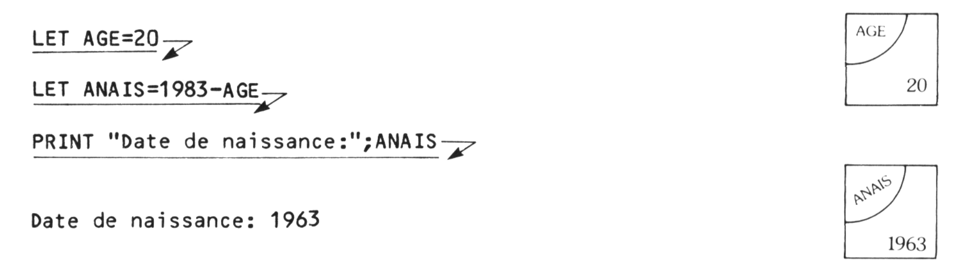
② **Lui ajoute 5**

③ **Range le total dans la case PX.**



Exemples :

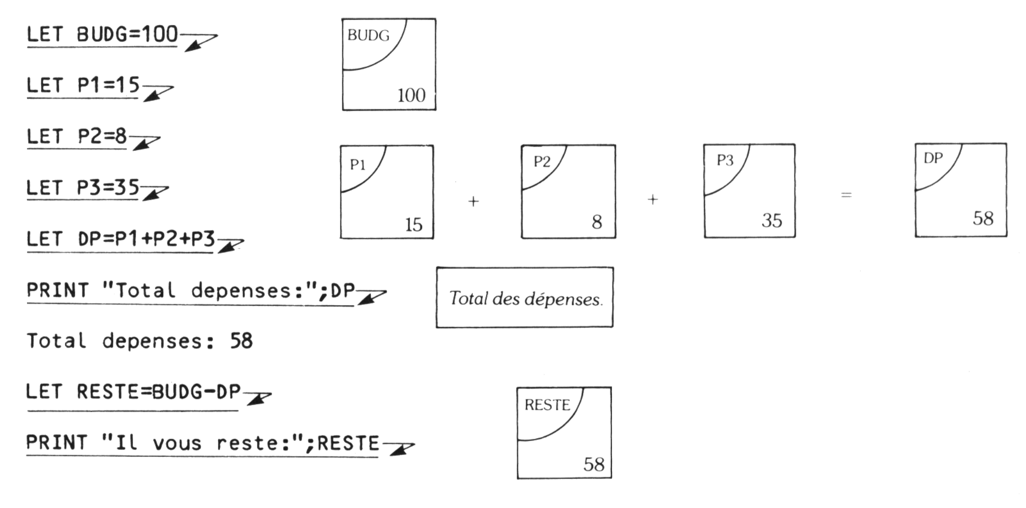
Calculons une date de naissance en fonction de l'âge :




```
LET AGE=20
LET ANAIS=1983-AGE
PRINT "Date de naissance:";ANAIS

Date de naissance: 1963
```

Calculons maintenant le total de dépenses (P1, P2, P3) et la somme qu'il reste :




```
LET BUDG=100
LET P1=15
LET P2=8
LET P3=35
LET DP=P1+P2+P3
PRINT "Total depenses:";DP
Total depenses: 58
LET RESTE=BUDG-DP
PRINT "Il vous reste:";RESTE

Il vous reste: 42
```

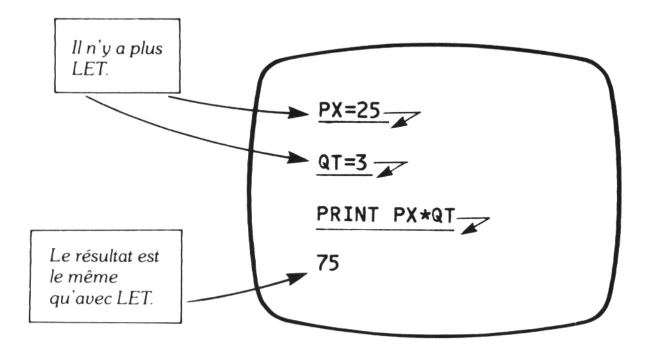
Pour vérifier les valeurs de P1, P2, P3, nous frappons :

```
PRINT P1,P2,P3
15    8     25
```

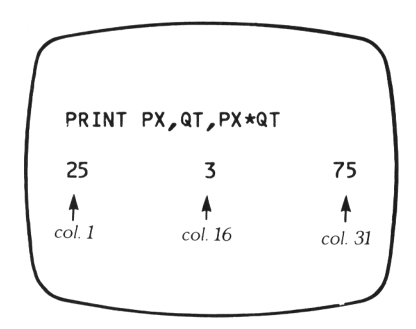
Si nous avons commis une erreur sur le prix de P2, il suffit de faire :

```
LET P2=18
LET DP=P1+P2+P3
PRINT "Total depenses:";DP
 Total depenses: 68
```

Pour affecter une valeur à une variable, nous avons utilisé l'instruction LET. Aujourd'hui, pour la plupart des BASICS, elle n'est plus obligatoire.



Pour afficher sur la même ligne le PRIX, la quantité et PRIX*QUANTITE, nous séparons les variables par des virgules.



Les résultats sont affichés en colonnes 1, 16 et 31 (de 15 en 15).
Le point virgule empêche le saut de ligne mais ne cadre pas les résultats qui sont édités les uns à la suite des autres, séparés par un espace.

```
PRINT PX;QT;QT*PX

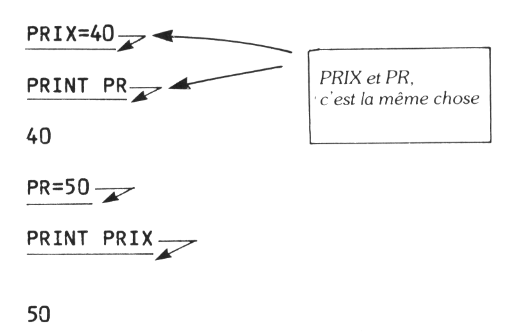
25  3  75
```

Remarques sur les noms de variables :

Le premier caractère doit obligatoirement être une lettre. Les autres caractères peuvent être indifféremment des lettres ou des chiffres.

A2 est bon
2A est interdit

Sur ORIC-1 seuls les deux premiers caractères sont pris en considération.



Pour BASIC, PRIX et PR sont une seule et même variable.

ATTENTION! ORIC 1 n'accepte pas de noms de variables comportant un mot clef du BASIC (voir annexe).

Par exemple TOTAL est interdit car TO est un mot clef du BASIC. En revanche TTAL est autorisé.

En bref

- *Une **VARIABLE** est une 'case' identifiable par un nom mnémonique, dans laquelle on range une valeur quelconque.*
- *L'instruction* **LET** *permet d'affecter une valeur à une variable.*
- *On peut ajouter une valeur à une variable X en faisant :*

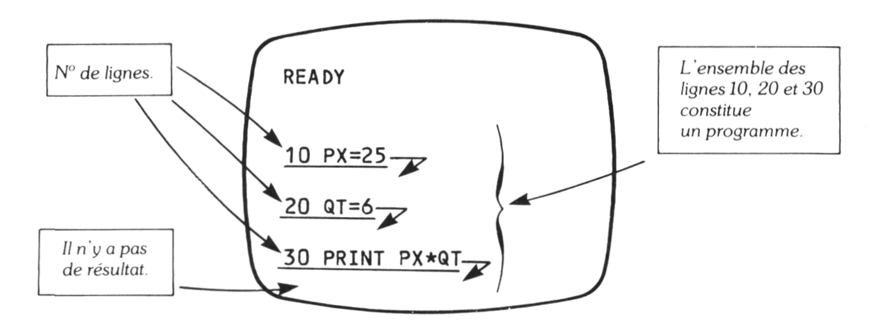
  LET X = X + valeur (ou X = X + valeur)

- *La **VIRGULE** permet d'afficher les résultats en colonnes 1, 16, 31...*
- *Le **POINT VIRGULE** ne cadre pas les résultats.*

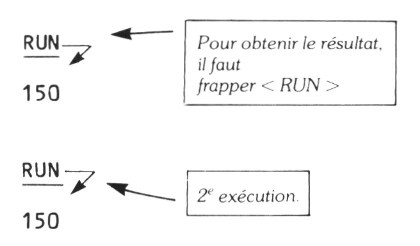
MÉMORISATION DES INSTRUCTIONS : PROGRAMME

Plutôt que de frapper plusieurs fois la même suite d'instructions, nous pouvons la mémoriser.

Pour mémoriser une suite d'instructions, nous les frappons en leur assignant un NUMERO de LIGNE (10, 20, 30,...)
Cette suite d'instructions s'appelle un PROGRAMME.



Nous constatons qu'il n'y a pas de résultat. **Pour obtenir un résultat, il faut demander l'EXECUTION du PROGRAMME par l'ordre RUN.**

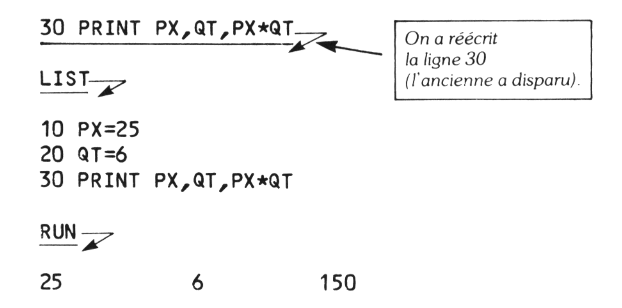


BASIC exécute les instructions une à une dans l'ordre de la numérotation. L'exécution du programme peut être demandée plusieurs fois.

Remarque : Les lignes n'ont pas à être entrées nécessairement dans l'ordre de la numérotation.

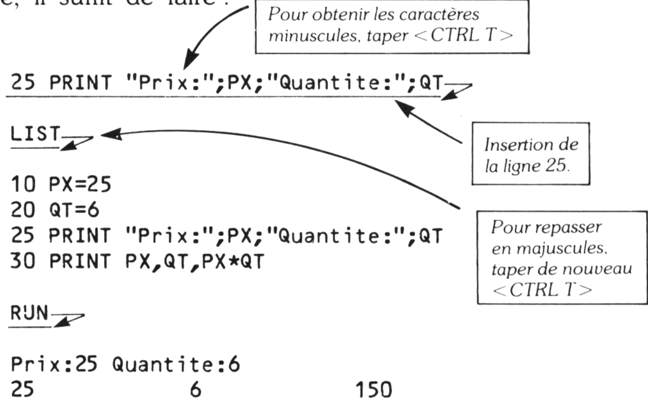
Modification d'une ligne :

Le plus simple est de la frapper à nouveau :



Ajout d'une ligne :

L'usage veut que la numérotation se fasse de 10 en 10. Pour 'insérer' une nouvelle ligne en 25 par exemple, il suffit de faire :



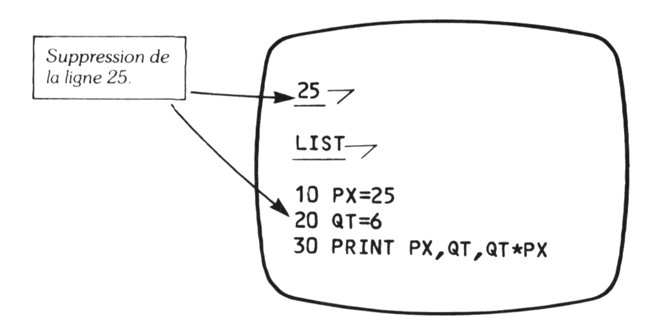
Liste d'un programme :

La commande LIST permet de lister le programme (ou une partie du programme) en mémoire à chaque fois que nous le voulons.

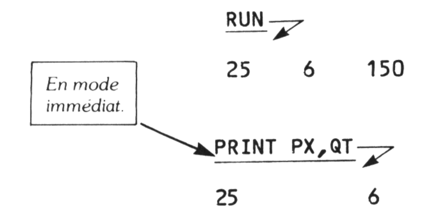
LIST 10 → liste la ligne 10
LIST – 30 → liste de la ligne 10 à 30
LIST 20 – → liste de la ligne 20 à la fin
LIST 20-40 → liste de la ligne 20 à la ligne 40

Suppression d'une ligne :

En frappant le numéro de ligne suivi de <RETURN>, la ligne est effacée.



A l'issue de l'exécution du programme (après le RUN), les variables PX et QT ont conservé leurs valeurs. On peut s'en assurer en frappant en mode immédiat (c'est-à-dire sans numéro de ligne) : PRINT PX, QT.



Pour connaître l'effet d'une instruction, il peut être plus rapide de frapper celle-ci en mode immédiat plutôt que de consulter un manuel.
De même, lors de la mise au point, le mode immédiat est très utile.
Aussi est-il essentiel d'acquérir le réflexe 'mode immédiat'.

En bref

- *Un **PROGRAMME** est une suite d'instructions numérotées.*

- *La numérotation des lignes se fait généralement de 10 en 10 afin de permettre l'insertion de nouvelles lignes.*

- *La commande* **RUN** *lance l'exécution d'un programme.*

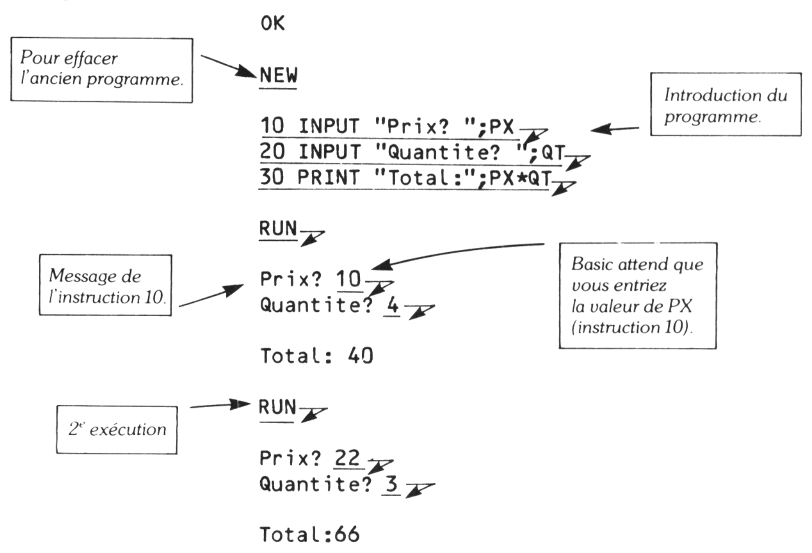
- *La commande* **LIST** *affiche à l'écran le programme présent en mémoire.*

POUR ENTRER DES VALEURS PENDANT L'EXÉCUTION : INPUT « Message ? » ; variable

L'instruction INPUT (ENTRER) permet de fournir des valeurs au programme PENDANT son EXECUTION par l'INTERMEDIAIRE du CLAVIER.

Lorsque BASIC rencontre une instruction INPUT pendant l'exécution du programme, il affiche à l'écran le message qui figure dans l'instruction INPUT (Prix ? sur l'exemple) puis ATTEND que l'opérateur ENTRE au clavier la valeur de la variable spécifiée dans cette instruction INPUT (PX sur l'exemple).

Lorsque l'opérateur a entré la valeur suivie de <RETURN>, l'exécution du programme se poursuit à la ligne suivante.



Remarques :

- A l'exécution, ORIC-1 place automatiquement un point d'interrogation après le message. Cependant, pour une meilleure lisibilité des programmes, nous avons volontairement ajouté ? dans le message. Naturellement, il n'est pas obligatoire.
- La commande NEW permet d'effacer l'ancien programme en mémoire. (Sur cet exemple, les nouvelles lignes effaceraient les anciennes).

L'instruction INPUT est très pratique puisqu'elle permet d'exécuter le programme pour différentes valeurs sans qu'il soit nécessaire de modifier le programme lui-même.

Exemple :

Calculons la consommation d'un véhicule.

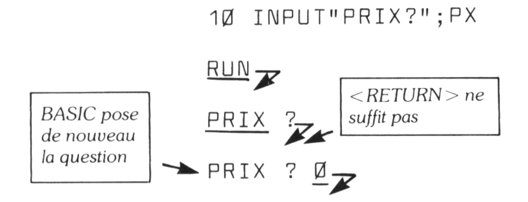
```
10 INPUT "Combien de litres? ";L
20 INPUT "Combien de Km   ? ";KM
30 :
40 C=(L/KM)*100
50 :
60 PRINT "Consommation aux 100 km:";C
70 :
80 GOTO 10

RUN

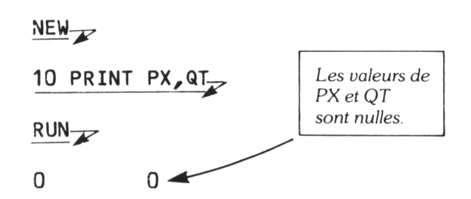
Combien de litres?  40
Combien de Km    ?  520

Consommation aux 100 km: 7.6
```

Remarque : Sur ORIC-1, lors de l'exécution d'une instruction INPUT, l'opérateur doit nécessairement entrer une valeur.
Appuyer sur <RETURN> ne suffit pas pour entrer une valeur nulle.



RUN initialise les variables à 0.



En bref

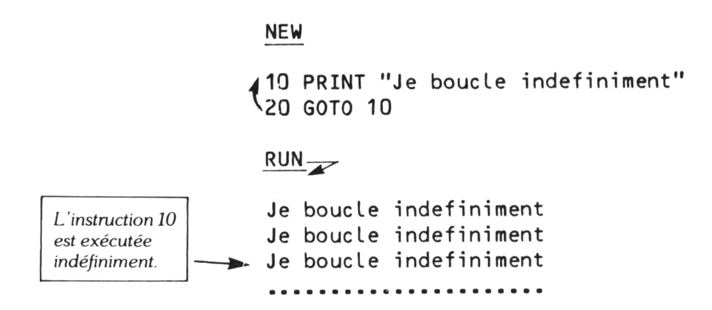
- *L'instruction* **INPUT** *permet d'introduire les valeurs des variables seulement au* ***MOMENT*** *de l'****EXECUTION*** *du* ***PROGRAMME.***

- *La commande* **NEW** *efface le programme en mémoire.*

BOUCLE : GOTO no de ligne

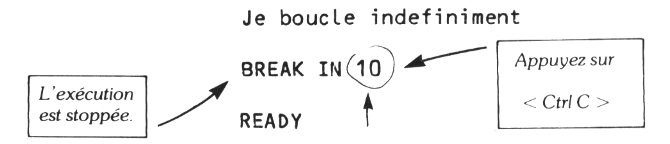
L'instruction 'GOTO no ligne' permet d'exécuter plusieurs fois une même suite d'instructions.

Montrons sur un exemple simple (sans utilité pratique) le mécanisme de GOTO :



BASIC exécute l'instruction 10 une première fois puis passe à l'exécution de l'instruction 20. Cette instruction '20 GOTO 10' signifie 'ALLER EN 10'. L'instruction 10 est donc exécutée à nouveau.

Naturellement, ce programme ne s'arrête jamais... sauf si vous appuyez sur la touche CTRL puis, tout en maintenant celle-ci, sur la touche C

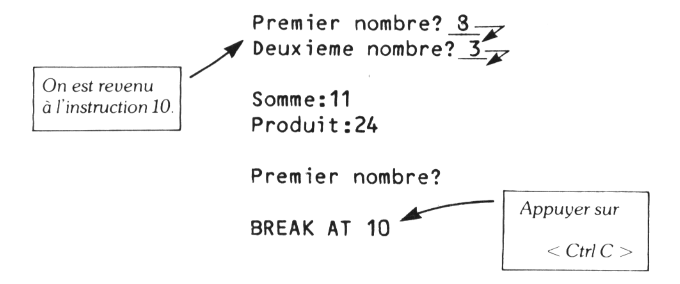


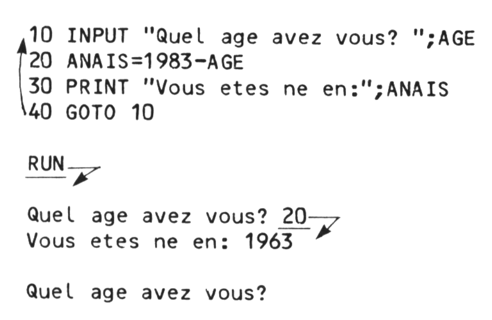
Utilisons cette instruction GOTO dans un contexte plus proche de la réalité. Supposons que nous devions faire plusieurs fois la somme de 2 nombres.

```
10 INPUT "Premier nombre? ";N1
20 INPUT "Deuxieme nombre?";N2
30 PRINT "Somme:";N1+N2
40 PRINT "Produit:";N1*N2
50 GOTO 10

RUN

Premier nombre? 5
Deuxieme nombre? 3
Somme:8
Produit:15
```





Exemple :

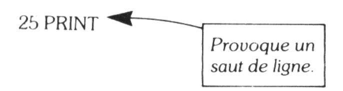
```
10 INPUT "Quel age avez vous? ";AGE
20 ANAIS=1983-AGE
30 PRINT "Vous etes ne en:";ANAIS
40 GOTO 10

RUN

Quel age avez vous? 20
Vous etes ne en: 1963

Quel age avez vous?
```

Pour provoquer un saut de ligne avant d'afficher « vous êtes né en : », ajouter l'instruction :



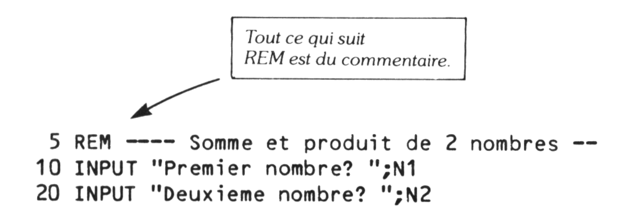
Remarque : Vous pouvez également taper 25 ?

A l'exécution le ? est remplacé par l'instruction PRINT

Instruction REM

Il ne s'agit pas d'une instruction à proprement parler puisqu'elle n'a pas d'influence sur le déroulement du programme.

Elle permet d'introduire des remarques dans le programme et d'améliorer sa présentation. (REM vient de REMark).




```
 5 REM ---- Somme et produit de 2 nombres --
10 INPUT "Premier nombre? ";N1
20 INPUT "Deuxieme nombre? ";N2
```

```
25 REM --
30 PRINT "Somme:";N1+N2
40 PRINT "Produit:";N1*N2
```

Pour séparer 2 phases du programme.

En bref

- *L'instruction* **GOTO no ligne** *(**ALLER EN** no ligne) provoque la poursuite de l'exécution du programme à la ligne spécifiée.*

- **REM** *permet d'introduire des commentaires dans le programme. Elle est ignorée à l'exécution.*

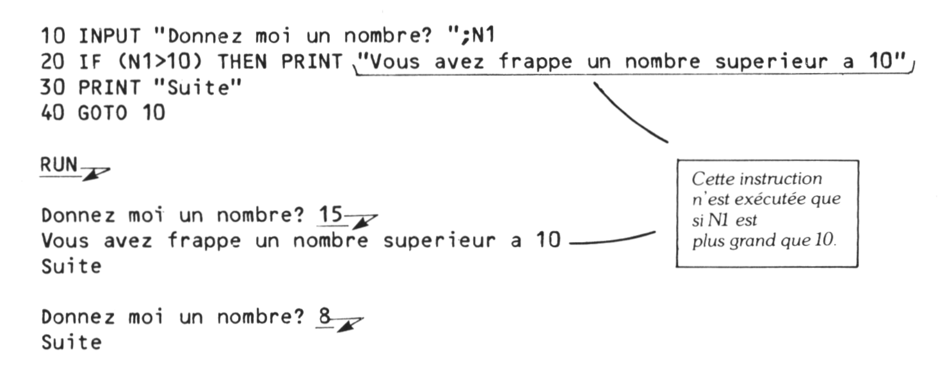
- *Afin de clarifier la lecture d'un programme, il est souhaitable d'y insérer des commentaires à l'aide de REM (ou ').*

- **<BREAK>** *ou* **<Ctrl C>** *permet de* ***STOPPER*** *l'exécution du programme.*

- *<CTRLC> permet de stopper l'exécution d'un programme*

TESTS :
IF condition THEN instruction
(SI condition ALORS instruction)

L'instruction IF... THEN... permet de tester si une condition est vraie et d'exécuter une ou plusieurs instructions si la condition est vraie.



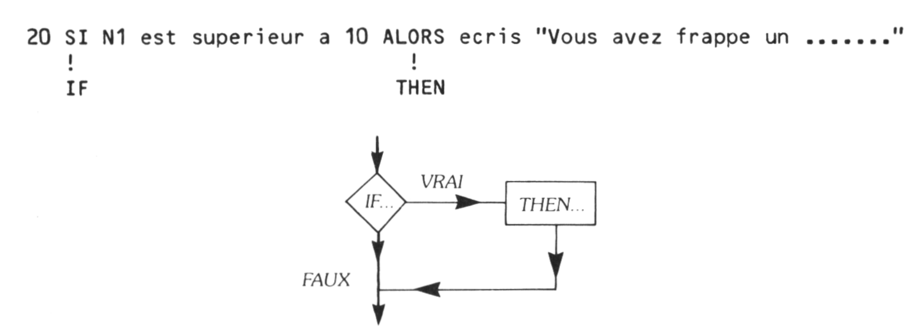
```
10 INPUT "Donnez moi un nombre? ";N1
20 IF (N1>10) THEN PRINT "Vous avez frappe un nombre superieur a 10"
30 PRINT "Suite"
40 GOTO 10

RUN

Donnez moi un nombre? 15
Vous avez frappe un nombre superieur a 10
Suite

Donnez moi un nombre? 8
Suite
```


L'instruction 20 signifie :

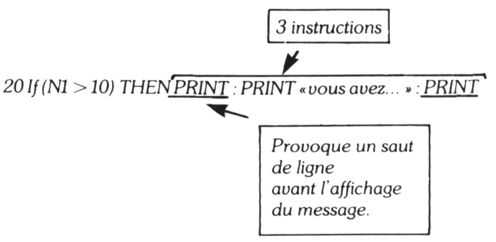


```
20 SI N1 est superieur a 10 ALORS ecris "Vous avez frappe un ......."
   !                            !
   IF                           THEN
```


Que l'instruction après THEN soit exécutée ou non, le programme se poursuit après l'instruction qui suit IF... THEN...

C'est dans cette instruction (ainsi que GOTO) que réside toute la puissance de l'ordinateur puisqu'elle rend possible l'exécution de séquences d'instructions seulement dans certains cas.

Remarque : Pour exécuter plusieurs instructions, il suffit de les séparer par le caractère : (deux points).

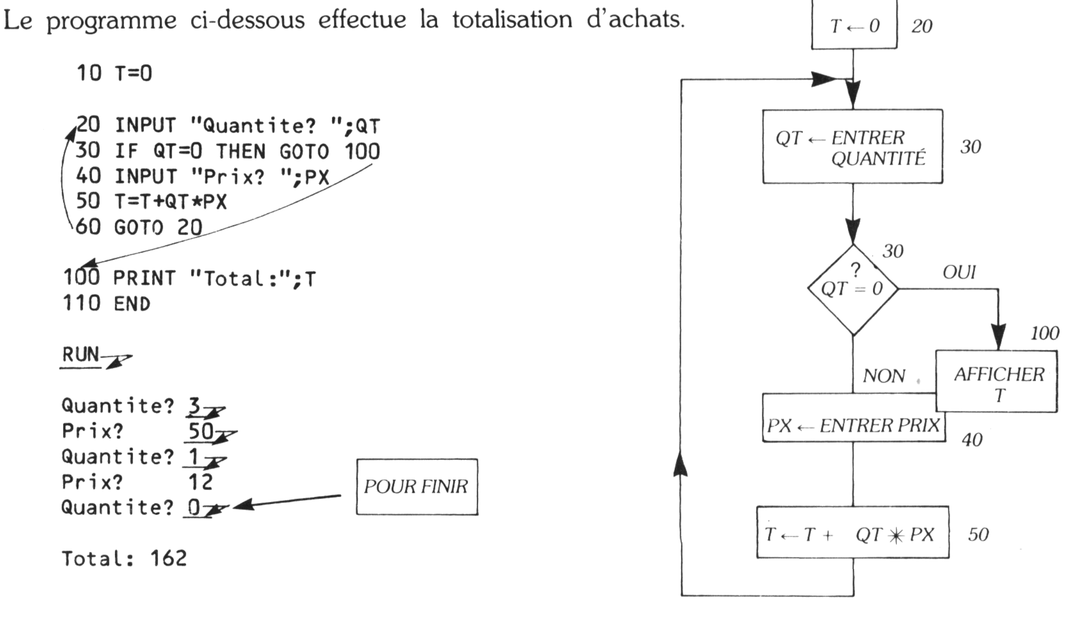


Les différents **opérateurs de comparaison** sont :
= égalité > supérieur à ⇒ supérieur ou égal à
< inférieur à <> différent de ⇐ inférieur ou égal à

Sortie de boucle :

L'instruction IF... THEN permet de « sortir » d'une boucle.

Le programme ci-dessous effectue la totalisation d'achats.

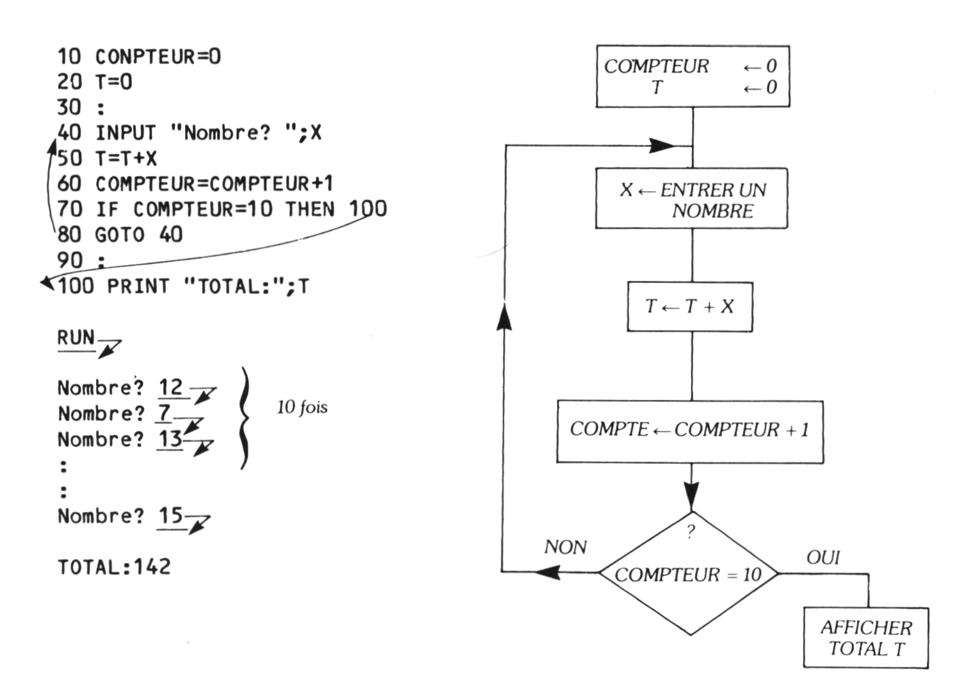


Lorsque tous les achats ont été totalisés, nous entrons 0 en réponse à la question 'Quantité ?'

Le test de la valeur 0 en 30 permet de 'sortir' de la boucle pour afficher le total.

Sortie de boucle à l'aide d'un compteur

Une autre façon pour sortir d'une boucle consiste à compter le nombre de passages dans la boucle à l'aide d'une variable (COMPTEUR sur l'exemple). Un test de la valeur de COMPTEUR permet de sortir de la boucle.




```
10 CONPTEUR=0
20 T=0
30 :
40 INPUT "Nombre? ";X
50 T=T+X
60 COMPTEUR=COMPTEUR+1
70 IF COMPTEUR=10 THEN 100
80 GOTO 40
90 :
100 PRINT "TOTAL:";T

RUN

Nombre? 12
Nombre? 7
Nombre? 13
:
:
Nombre? 15

TOTAL:142
```

Si le nombre de valeurs à ajouter n'est pas connu à l'avance il suffit de faire :

```
5 INPUT "Combien de nombres? ";NB

70 IF COMPTEUR=NB THEN 110


RUN

Combien de nombres? 4

Nombre? 12
Nombre? 6
Nombre? 8
Nombre? 13

TOTAL: 39
```

4 fois

Test de conditions multiples :

Des conditions multiples peuvent être testées avec les OPERATEURS LOGIQUES AND et OR.

condition 1 AND condition 2 : Les instructions après THEN ne sont exécutées que si condition 1 ET condition 2 sont vraies.

```
10 INPUT "Donnez un nombre? ";N
20 IF (N>0) AND (N<10) THEN PRINT "Ce nombre est superieur a 0 ET
                                          inferieur a 10"
30 GOTO 10

RUN

Donnez un nombre? 5
Ce nombre est superieur a 0 ET inferieur a 10
Donnez un nombre? 13
Donnez un nombre?
```

condition 1 OR condition 2 : Les instructions après THEN sont exécutées si condition 1 OU condition 2 est vraie (ou les deux).

```
10 INPUT "Donnez 2 nombres X,Y ? ";X,Y
20 IF (X>0) OR (Y>0) THEN PRINT "X OU Y est positif (ou les deux)"
30 GOTO 10

RUN

Donnez 2 nombres X,Y? 12,4
X OU Y est positif (ou les deux)

Donnez 2 nombres X,Y? -4,-1
```

IF condition THEN instructions ELSE instructions

IF... THEN... ELSE... permet d'exécuter des instructions si la condition testée est fausse

```
10 INPUT "Quel est votre age? ";AGE
20 IF AGE=>18 THEN PRINT "Vous etes majeur" ELSE PRINT "Vous etes mineur"
30 PRINT "suite"
40 GOTO 10
```

Instruction exécutée si la condition testée est vraie (PRINT "Vous etes majeur")

Instruction exécutée si la condition testée est fausse (PRINT "Vous etes mineur")

```
RUN

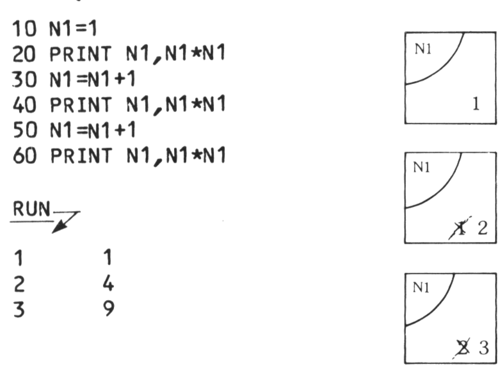
Quel est votre age? 25
Vous etes majeur
suite

Quel est votre age? 17
Vous etes mineur
suite
```

BOUCLE AUTOMATIQUE : FOR... NEXT

Soit à afficher les nombres 1, 2, 3 et leurs carrés 1*1, 2*2, 3*3.

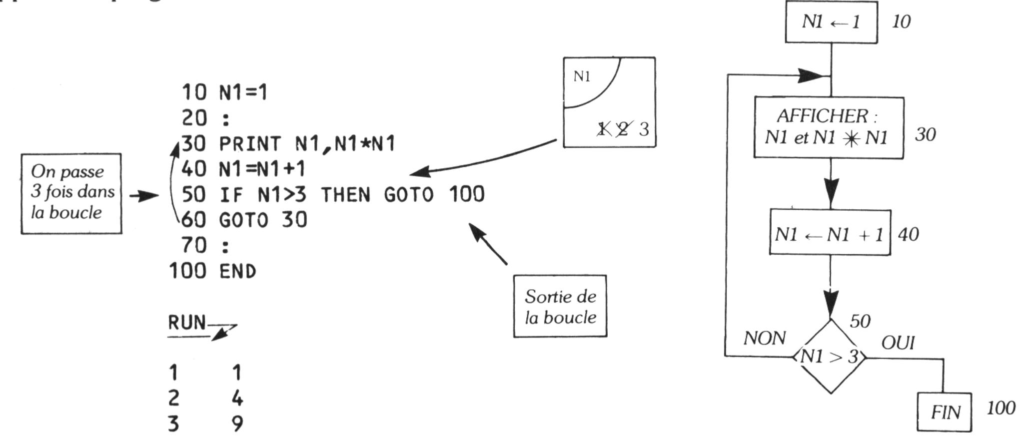
1re solution : (sans GOTO et IF... THEN)



Pour calculer les carrés des nombres de 1 à 100, cette méthode serait fastidieuse.

2e solution : (avec GOTO et IF... THEN...)

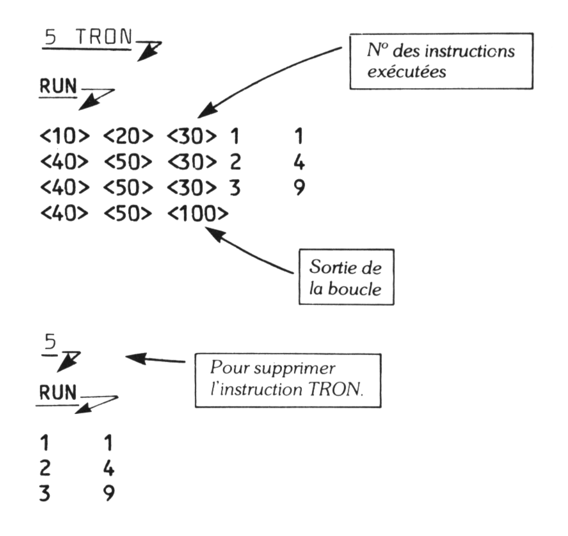
Aussi, plutôt que d'écrire plusieurs fois la séquence :
N1 = N1 + 1
PRINT N1, N1*N1
écrivons-la une seule fois.
Pour l'exécuter plusieurs fois, nous utilisons une instruction de branchement GOTO. et pour 'sortir' de la boucle, nous testons la valeur de N1. Dès que N1 devient supérieur à 3, nous stoppons le programme.



Pour obtenir la liste des carrés des nombres de 1 à 100, il suffit de changer l'instruction 50 :

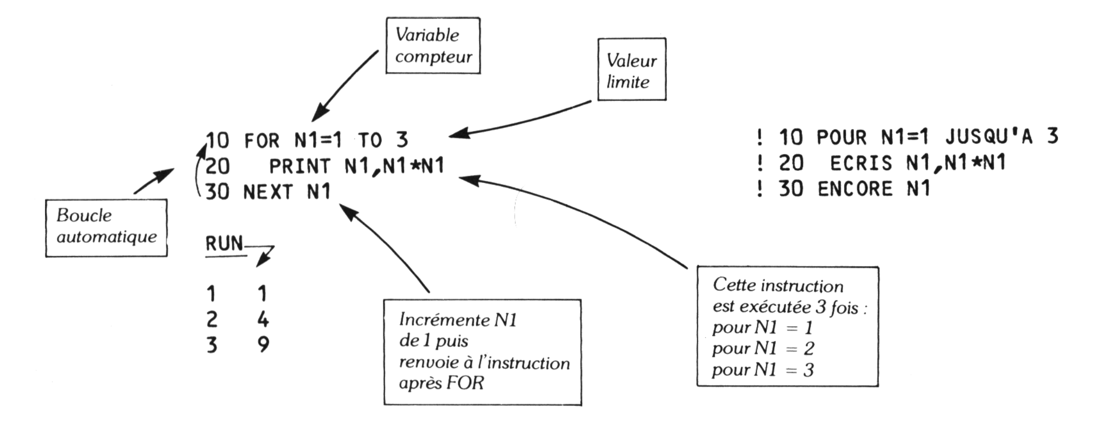
```
50 IF N1=100 THEN GOTO 100
```

Pour bien voir comment se 'déroule' le programme, utilisons l'instruction 'TRON' (TRACE ON). Les numéros des instructions exécutées sont alors visualisées.



3ᵉ solution : (boucle automatique FOR... NEXT...)

Simplifions l'écriture du programme précédent avec une boucle 'automatique', la boucle FOR... NEXT.



Comment se déroule ce programme ?

— L'instruction 10 stocke en mémoire la 'valeur limite' 3 et donne à la 'variable compteur' N1 la valeur 1

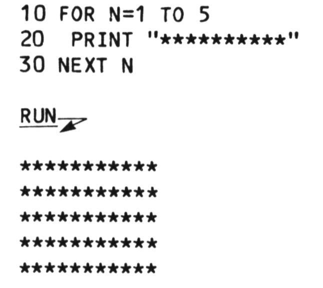
— L'instruction 20 est exécutée avec N1 = 1

— L'instruction '30 NEXT N1' augmente N1 de 1 et teste si N1 est INFERIEUR ou EGAL à la 'valeur limite' 3.
Si N1 est inférieur ou égal à 3, l'instruction après FOR (20) est à nouveau exécutée avec N1 = 2

Ainsi, l'instruction 20 est exécutée 3 fois avec N1 = 1, 2, 3.

Exemples :

Pour afficher 5 lignes d'étoiles :



```
10 FOR N=1 TO 5
20  PRINT "**********"
30 NEXT N

RUN

***********
***********
***********
***********
***********
```

Pour afficher les carrés des nombres de 4 à 6 :

```
10 FOR N1=4 TO 6
20  PRINT N1,N1*N1
30 NEXT N1
40 PRINT "La boucle est terminee"

RUN

4   16
5   25
6   36
La boucle est terminee

 5 TRON

RUN

<10> <20> 4   16
<30> <20> 5   25
<30> <20> 6   36
<30> <40> La boucle est terminee

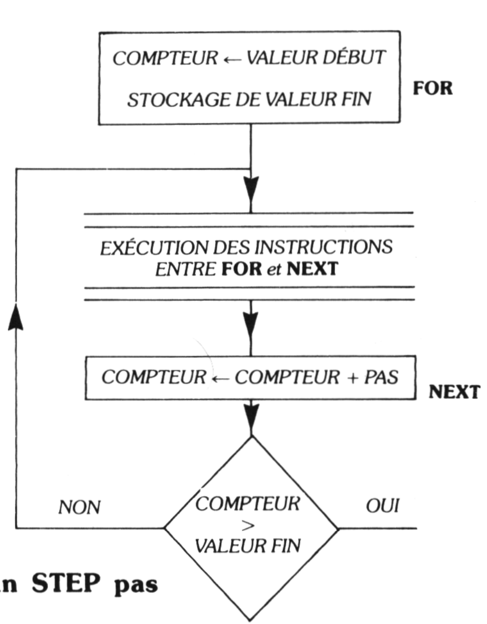
 5
```

Syntaxe complète de la boucle FOR :

Un pas d'exécution peut être spécifié par STEP :

La spécification 'STEP pas' indique qu'il faut faire progresser le compteur de 'pas' à chaque exécution de la boucle.

Si l'on omet la spécification STEP, le compteur progresse, par défaut, de 1 à chaque passage. Ne pas spécifier le 'pas' ou lui donner une valeur égale à 1 est donc identique.



FOR compteur = valeur début TO valeur fin STEP pas
instruction 1
instruction 2
....
NEXT compteur

Lorsque l'instruction FOR est exécutée, BASIC affecte à la 'variable compteur' la 'valeur début' spécifiée et stocke la 'valeur limite' indiquée.
Toutes les instructions entre FOR et NEXT sont d'abord exécutées avec compteur = valeur début.
L'exécution de l'instruction NEXT augmente la valeur de 'compteur' de 'pas'

- Si la valeur de 'compteur' est inférieure à la 'valeur limite', les instructions entre FOR et NEXT sont à nouveau exécutées avec la nouvelle valeur de 'compteur'.
- Si la valeur de 'compteur' a atteint la 'valeur fin', l'exécution de la boucle s'achève et le programme se poursuit après l'instruction NEXT.

Exemples :

STEP positif :

```
10 FOR C=1 TO 5 STEP 2
20  PRINT C;
30 NEXT C
40 PRINT "C'est fini"

RUN

1  3  5  C'est fini
```

Table des sinus :

```
10 FOR ANG=0 TO 3.14 STEP 3.14/10
20   PRINT ANG,SIN(ANG)
30 NEXT ANG

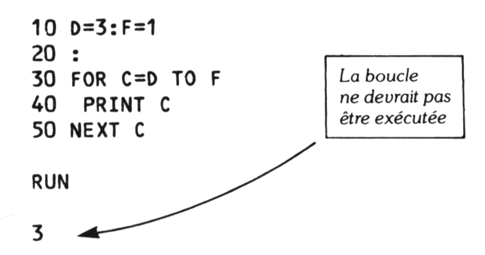
RUN

 0       0
.314    .308866
.628    .587528
.942    .808736
```

STEP négatif :

```
10 FOR C=3 TO 1 STEP-1
20 PRINT C;
30 NEXT C
RUN
3 2 1
```

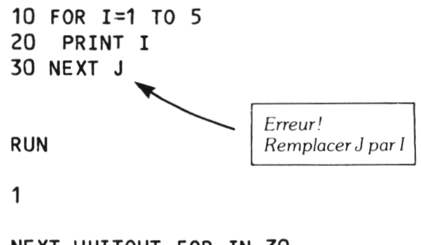
Une boucle FOR est exécutée au moins une fois, même si 'valeur limite' est inférieure à 'valeur début'.



Il faut donc prévoir un test.

```
35  IF C>F THEN 50
```

Erreurs :




```
10 FOR I=1 TO 5
20  PRINT I
30 NEXT J

RUN

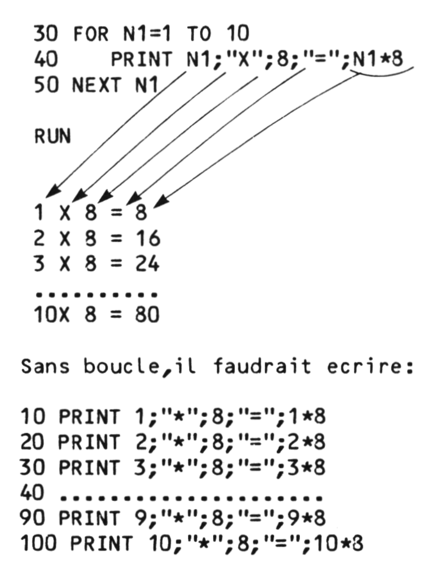
1

NEXT WHITOUT FOR IN 30
```

Exemple :

Table de multiplication par 8 :

Cette boucle FOR affiche la table de multiplication par 8.



```
30 FOR N1=1 TO 10
40    PRINT N1;"X";8;"=";N1*8
50 NEXT N1

RUN

1 X 8 = 8
2 X 8 = 16
3 X 8 = 24
..........
10X 8 = 80
```

```
Sans boucle,il faudrait ecrire:

10 PRINT 1;"*";8;"=";1*8
20 PRINT 2;"*";8;"=";2*8
30 PRINT 3;"*";8;"=";3*8
40 ......................
90 PRINT 9;"*";8;"=";9*8
100 PRINT 10;"*";8;"=";10*8
```

Ce qui serait, vous en conviendrez, par trop fastidieux!

Pour généraliser ce programme à toutes les tables de multiplication :

```
10 INPUT "Quelle table? ";T
20 :
30 FOR N1=1 TO 10
40  PRINT N1;"X";T;"=";N1*T
50 NEXT N1
60 GOTO 10

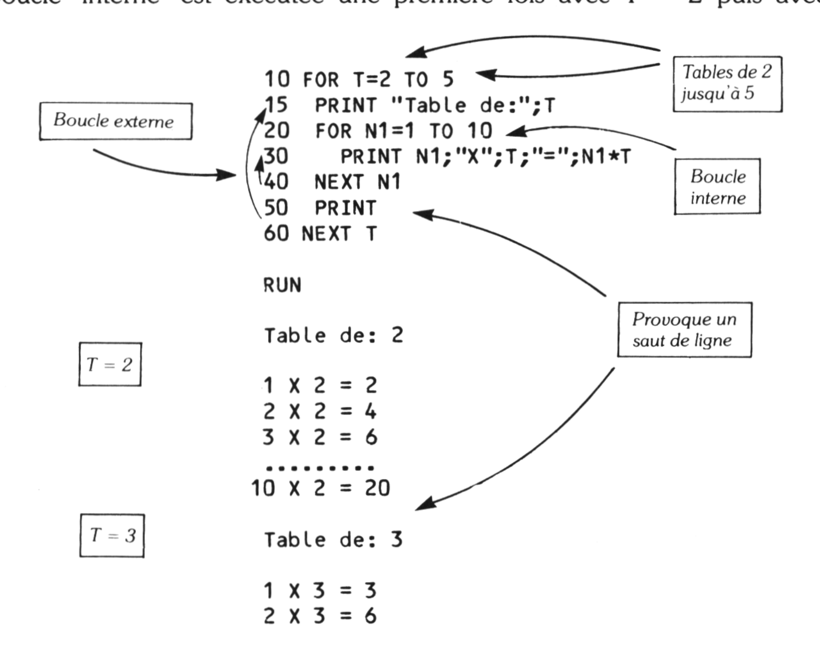
RUN

Quelle table? 9

1 X 9 = 9
2 X 9 = 18
```

Boucles emboîtées :

Plusieurs boucles FOR peuvent être emboîtées.
Le programme ci-dessous affiche les tables de multiplication de 2 à 5.
La boucle 'interne' est exécutée une première fois avec T = 2 puis avec T = 3 etc.




```
10 FOR T=2 TO 5
15  PRINT "Table de:";T
20  FOR N1=1 TO 10
30    PRINT N1;"X";T;"=";N1*T
40  NEXT N1
50  PRINT
60 NEXT T

RUN

Table de: 2

1 X 2 = 2
2 X 2 = 4
3 X 2 = 6
.........
10 X 2 = 20

Table de: 3

1 X 3 = 3
2 X 3 = 6
```

Essayez ce programme en y insérant l'instruction TRON pour mieux comprendre son déroulement.

Exemple :

Voici une horloge qui affiche les heures, minutes et secondes : La temporisation en ligne 50 doit être adaptée à votre ordinateur.

```
10 FOR H=0 TO 23
20  FOR M=0 TO 59
30    FOR S=0 TO 59
40      PRINT H;"h.";M;"m.";S;"s."
50      FOR TP=1 TO 170:NEXT TP            :REM temporisation
60    NEXT S
70  NEXT M
80 NEXT H

RUN

0 h. 0 m. 0 s.
0 h. 0 m. 1 s.
0 h. 0 m. 2 s.
```

Vous pouvez obtenir l'affichage 'direct' en faisant :

```
5Ø PLOT 1Ø,1Ø,STR$(H)+"H"
52 PLOT 15,1Ø,STR$(M)+"M"
54 PLOT 21,1Ø,STR$(S)+"S"
```

(voir chapitre 'Adressage direct écran)

REPEAT... UNTIL

Cette instruction permet d'exécuter des instructions jusqu'à ce qu'une condition soit vérifiée.

```
10 REPEAT
20 :PRINT "APPUYEZ SUR UNE TOUCHE"
30 UNTIL KEY$<>""
40 :
50 PRINT C'EST FINI
RUN

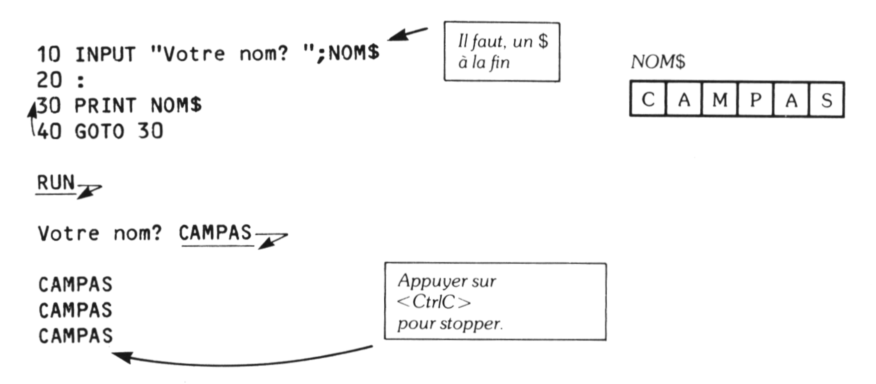
APPUYEZ SUR UNE TOUCHE
APPUYEZ SUR UNE TOUCHE
```

Dès que vous appuyez sur une touche du clavier, la boucle **REPEAT... UNTIL** s'arrête.

LES CHAÎNES DE CARACTÈRES

Il existe des variables du type 'chaîne de caractères'. Elles sont distinguées des variables numériques par la présence du signe $ à la fin du nom des variables.

Ces variables sont utilisées dans la plupart des applications (gestion, éducation, jeux).



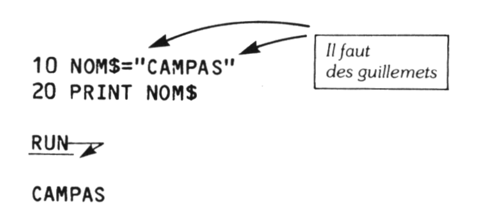
Pour empêcher le saut de ligne après l'édition du nom, il suffit d'ajouter un point virgule après NOM$.

```
30 PRINT NOM$;

RUN

CAMPASCAMPASCAMPAS
```

L'affectation de la valeur 'CAMPAS' à NOM$ nécessite la présence de guillemets. Autrement 'CAMPAS' serait considéré comme un nom de variable.

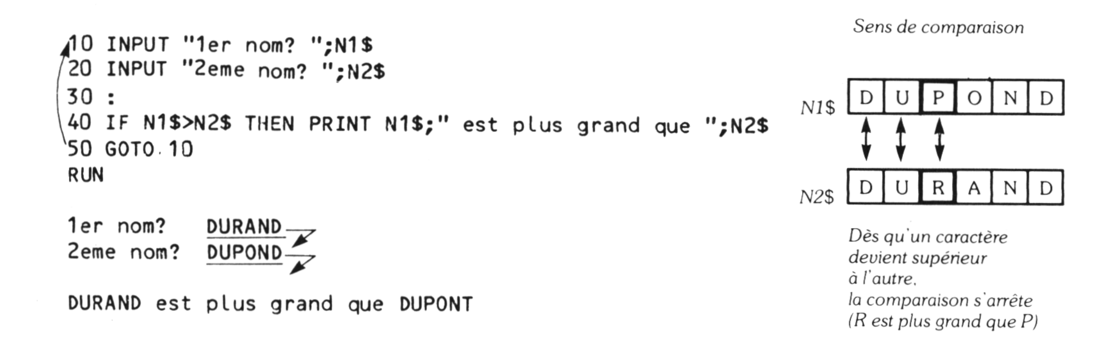


La longueur des chaînes est limitée à 255 caractères.

Comparaison de chaînes :

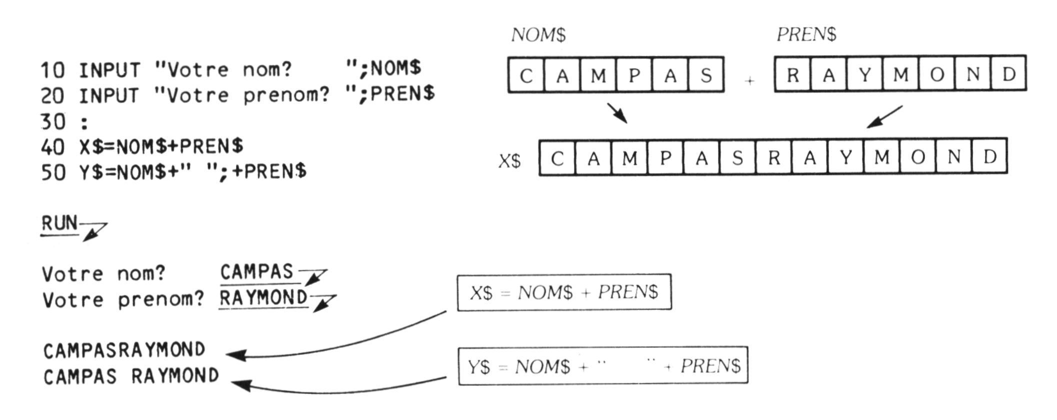
La comparaison de chaînes est faite (par BASIC) de gauche à droite caractère par caractère.

Dès qu'un caractère d'une chaîne devient différent du vis à vis, la comparaison s'arrête.



Concaténation de chaînes :

La concaténation (ou addition) **de chaînes se fait à l'aide de l'opérateur** '+' (comme pour les variables numériques).



Longueur d'une chaîne : LEN(chaîne)

LEN (chaîne) donne le nombre de caractères d'une chaîne.

```
10 INPUT "Votre nom? ";NOM$
20 L=LEN(NOM$)
30 PRINT "Votre nom comporte ";L;"caracteres"

RUN

Votre nom? CAMPAS

Votre nom comporte 6 caracteres.
```


Une variable chaîne peut voir sa longueur varier en cours d'exécution de programme. La longueur maximum n'a pas à être réservée en début de programme.

LEFT$(chaîne, longueur à prendre)

Donne les caractères de gauche d'une chaîne.

```
10 X$="ORIC1POUR TOUS"
20 Y$=LEFT$(X$,5)
30 PRINT Y$

RUN

BASIC
```


Par exemple :

```
10 INPUT "Votre nom? ";NOM$
20 FOR L=1 TO LEN(NOM$)
30   PRINT LEFT$(NOM$,L)
40 NEXT L

RUN

Votre nom?  CAMPAS

C
CA
CAM
CAMP
CAMPA
CAMPAS
```

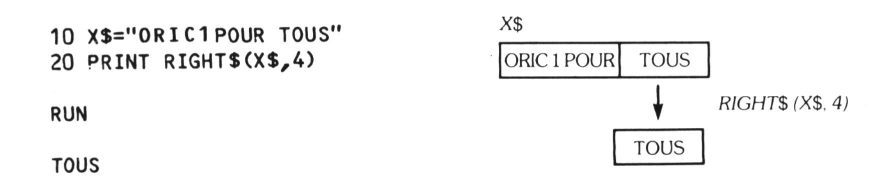
RIGHT$(chaîne, longueur à prendre)

Donne les caractères de droite d'une chaîne.

```
10 X$="ORIC1POUR TOUS"
20 PRINT RIGHT$(X$,4)

RUN

TOUS
```

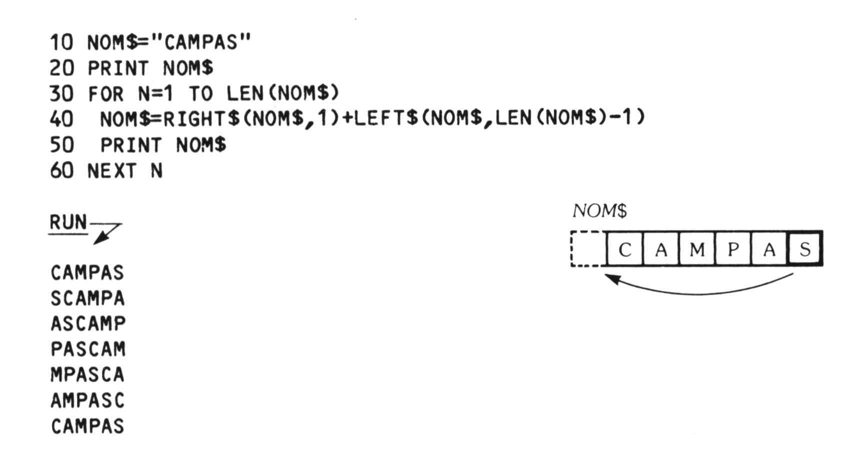


Le programme ci-dessous affiche les permutations circulaires d'une chaîne :

```
10 NOM$="CAMPAS"
20 PRINT NOM$
30 FOR N=1 TO LEN(NOM$)
40  NOM$=RIGHT$(NOM$,1)+LEFT$(NOM$,LEN(NOM$)-1)
50  PRINT NOM$
60 NEXT N

RUN

CAMPAS
SCAMPA
ASCAMP
PASCAM
MPASCA
AMPASC
CAMPAS
```



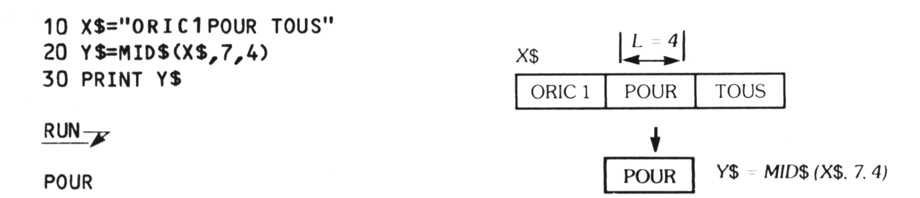
MID$(chaîne, position début, longueur à prendre)

Donne les caractères du milieu d'une chaîne.

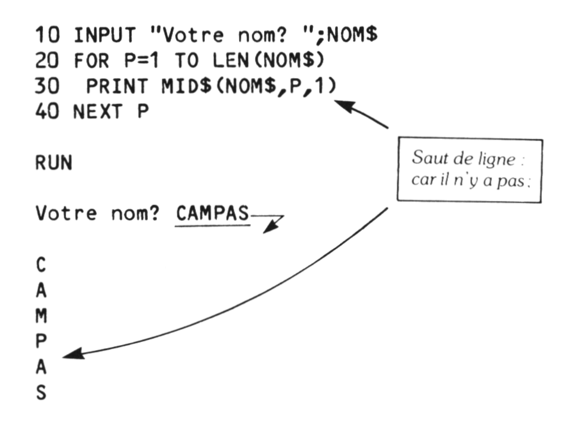
```
10 X$="ORIC1POUR TOUS"
20 Y$=MID$(X$,7,4)
30 PRINT Y$

RUN

POUR
```



Par exemple :




```
10 INPUT "Votre nom? ";NOM$
20 FOR P=1 TO LEN(NOM$)
30  PRINT MID$(NOM$,P,1)
40 NEXT P

RUN

Votre nom? CAMPAS

C
A
M
P
A
S
```

Le programme ci-dessous affiche le nom à l'envers.

```
10 INPUT "Votre nom? ";NOM$
20 FOR P=LEN(NOM$) TO 1 STEP-1
30  PRINT MID$(NOM$,P,1);
40 NEXT P

RUN

Votre nom? ZDUNECK

KCENUDZ
```

Celui-ci découpe une phrase :

```
10 INPUT "Entrez une phrase? ";PH$
20 :
30 FOR P=1 TO LEN(PH$)
40  X$=MID$(PH$,P,1)
50  IF X$<>" " THEN Y$=Y$+X$
60  IF X$=" " THEN PRINT Y$:Y$=""
70 NEXT P
80 PRINT Y$

RUN

Entrez une phrase?  LE PETIT CHAT RONRONNE

LE
PETIT
CHAT
RONRONNE
```

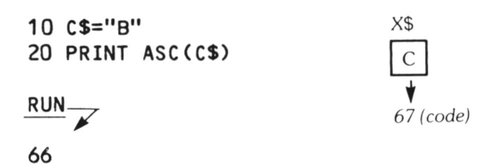
POUR REMPLACER UNE PARTIE DE CHAINE

```
 10 X$="AAAAAAAAA"
 20 P=4:Y$="ZZZ"
 30 GOSUB 100
 40 PRINT X$
 50 END
 60 :
100 X$=LEFT$(X$,P-1)+Y$+RIGHT$(X$,LEN(X$)-P-LEN(Y$)+1)
110 RETURN
```

ASC(caractère)

Donne le code d'un caractère.

Tous les caractères sont représentés de façon interne (pour la machine) sous forme binaire. Le programmateur a accès à ces codes sous forme décimale.

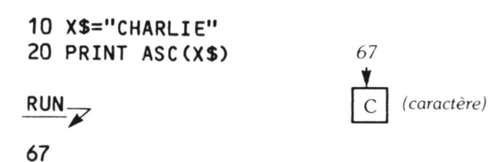


Les codes de l'alphabet (A, B, C,... Z) sont 65, 66, 67... 91

ASC(chaîne)

Donne le code du premier caractère de la chaîne.

La chaîne ne doit pas être vide.



CHR$(code)

Fournit un caractère. Certains caractères seulement sont imprimables.

```
10 X=67
20 PRINT CHR$(X)

RUN

C

10 FOR C=65 TO 65+26
20  PRINT CHR$(C);
30 NEXT C

RUN

ABCD.....Z
```

Ce programme fait correspondre à chaque lettre de l'alphabet, la lettre suivante (par exemple, à A on fait correspondre B)

```
10 INPUT "Votre nom? ";NOM$
20 :
30 FOR P=1 TO LEN(NOM$)
40  X=ASC(MID$(NOM$,P,1))
50  PRINT CHR$(X+1);
60 NEXT P

RUN

Votre nom? CAMPAS

DBNQBT  <----  Nom codé
```

Certains caractères servent à envoyer des commandes aux périphériques :

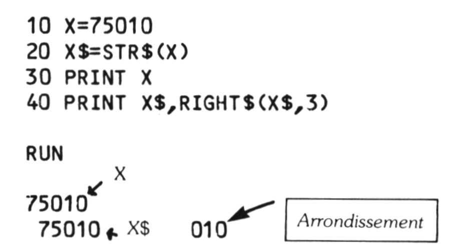
Quelques codes ASCII :

PRINT CHR$(10); provoque un saut de ligne (sans retour en début de ligne)
PRINT CHR$(13); provoque un retour en début de ligne (sans saut de ligne)

La fonction CHR$ (code) est utilisée pour l'envoi de caractères spéciaux à l'écran : changement de couleur, clignotement, doublement de la taille des caractères, etc. (cf caractères spéciaux p. 133).

STR$(X)

Convertit un nombre sous forme d'une chaîne, permettant ainsi l'accès à chacun des chiffres par les fonctions LEFT$, RIGHT$, MID$.




```
10 X=75010
20 X$=STR$(X)
30 PRINT X
40 PRINT X$,RIGHT$(X$,3)

RUN

75010
 75010      010
```

Notez la présence d'un caractère en début de chaîne. Il correspond à la place du signe + (implicite). Son code est 2.

VAL(chaîne)

Donne la valeur numérique d'une chaîne commençant par des chiffres (ou un espace).

Si la chaîne commence par une lettre, la valeur est nulle.

```
10 PRIX$="22 DOLLARS"
20 X=VAL(PRIX$)
30 PRINT X

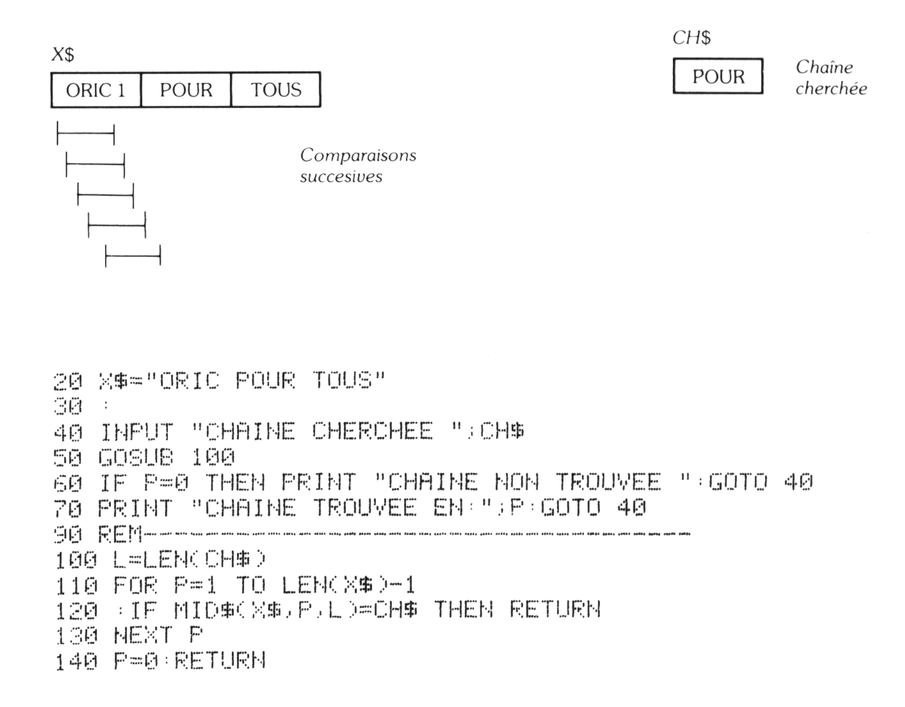
RUN

22
```

Une chaîne ne peut être traitée directement comme une valeur

```
10 PRIX$="22"
20 PRINT PRIX$*3      ---> provoque une erreur
```

RECHERCHE D'UNE CHAINE DANS UNE AUTRE CHAINE :




```
20 X$="ORIC POUR TOUS"
30 :
40 INPUT "CHAINE CHERCHEE ";CH$
50 GOSUB 100
60 IF P=0 THEN PRINT "CHAINE NON TROUVEE ":GOTO 40
70 PRINT "CHAINE TROUVEE EN:";P:GOTO 40
90 REM-----------------------------------------------
100 L=LEN(CH$)
110 FOR P=1 TO LEN(X$)-1
120 :IF MID$(X$,P,L)=CH$ THEN RETURN
130 NEXT P
140 P=0:RETURN
```

Remarque : Une chaîne entrée par INPUT ne doit pas comporter de virgule (qui est considérée comme séparateur).

```
10 INPUT "Rue? ";RUE$

RUN

Rue? 11 rue de CLATIGNY
```

Si l'opérateur entre une virgule après 11, tout ce qui suit 11 est ignoré.

MISE AU POINT DES PROGRAMMES : CTRL C-STOP-CONT

Les programmes ne fonctionnent pas toujours 'du premier coup'. Basic envoie des messages pour certaines erreurs (de syntaxe par exemple) mais ne détecte pas les erreurs de logique. Pour les cas les plus délicats, il faut suivre le déroulement du programme étape par étape, ce qui est relativement simple en BASIC.

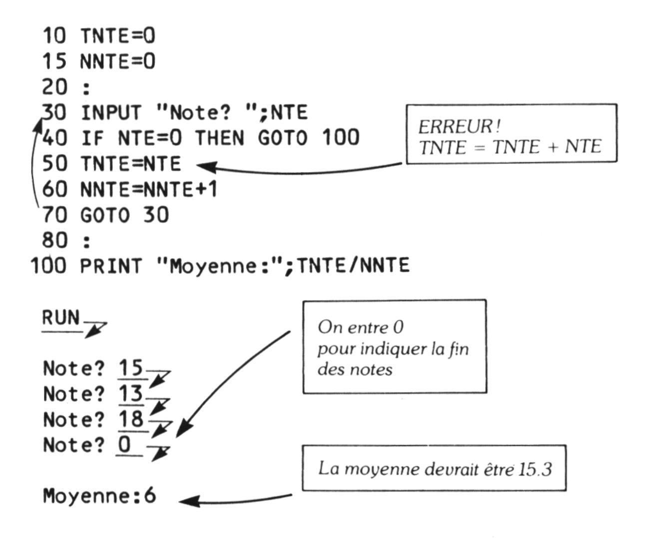
- En appuyant sur <CTRL C>, nous interrompons l'exécution du programme.

Nous pouvons alors visualiser les valeurs des variables en mode immédiat.

L'exécution interrompue peut être poursuivie en frappant CONT (continue).

Exemple

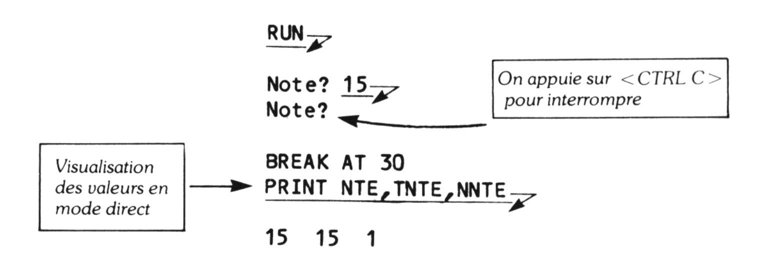
Le programme ci-dessous effectue la moyenne de plusieurs notes.
Nous avons commis (volontairement !) une erreur : En 50, au lieu de TNTE = TNTE + NTE, nous avons écris TNTE = NTE.



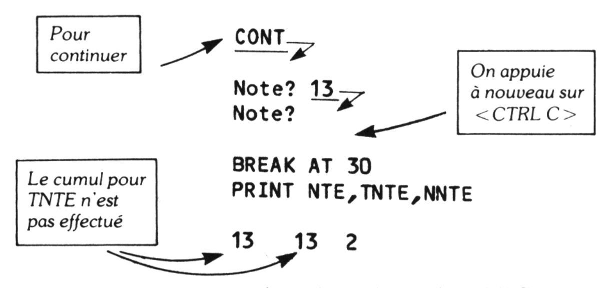
Naturellement, la moyenne obtenue (6) est fausse.

Remarque : Les instructions 10 et 15 qui initialisent les valeurs de TNTE et NNTE à zéro ne sont pas indispensables puisque RUN les initialise à zéro. Il est cependant plus prudent de le faire dans un programme plus important où TNTE et NNTE pourraient déjà avoir été utilisées dans une autre partie de programme et avoir une valeur non nulle.

Exécutons à nouveau le programme et interrompons-le après avoir entré la première note. Nous pouvons visualiser en mode direct les valeurs des variables NTE, TNTE et NNTE.



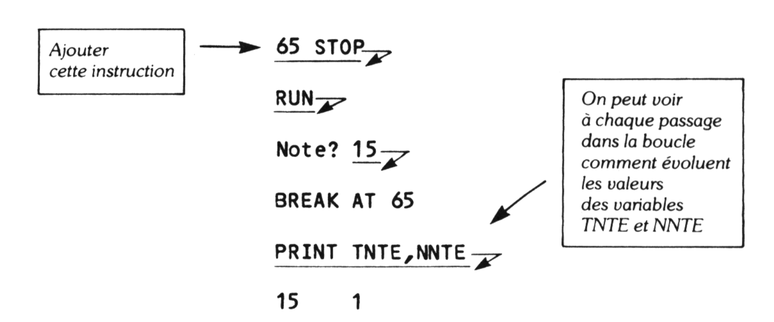
Pour l'instant, rien d'anormal.
Frappons CONT pour continuer l'exécution du programme et interrompons à nouveau le programme après avoir entré la 2[e] note :



Nous nous apercevons en regardant la valeur de TNTE que le cumul des notes n'est pas effectué.

STOP :

Au lieu d'appuyer sur les touches <CTRL C>, nous aurions pu placer une instruction 'STOP' en 65.



```
CONT

Note? 13

BREAK AT 65
PRINT TNTE,NNTE

13   2

CONT
```

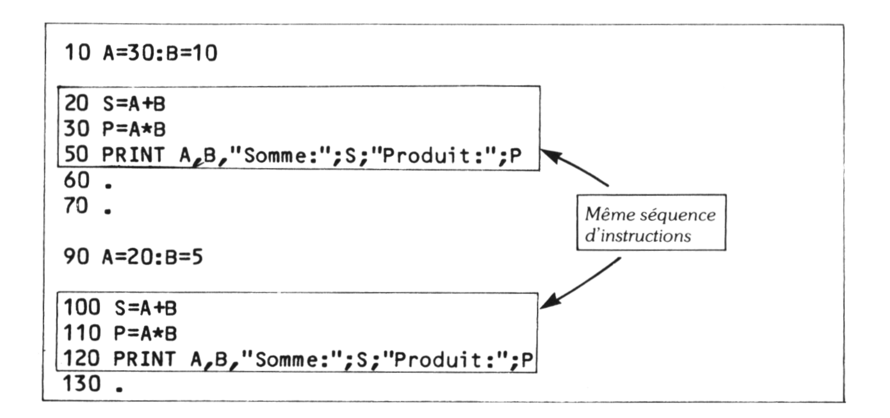
En cas de 'BOUCLAGE' de programme :

Lorsqu'un programme ne s'arrête pas, ajouter l'instruction TRON en début de programme. On peut ainsi localiser la partie du programme où la boucle s'effectue.

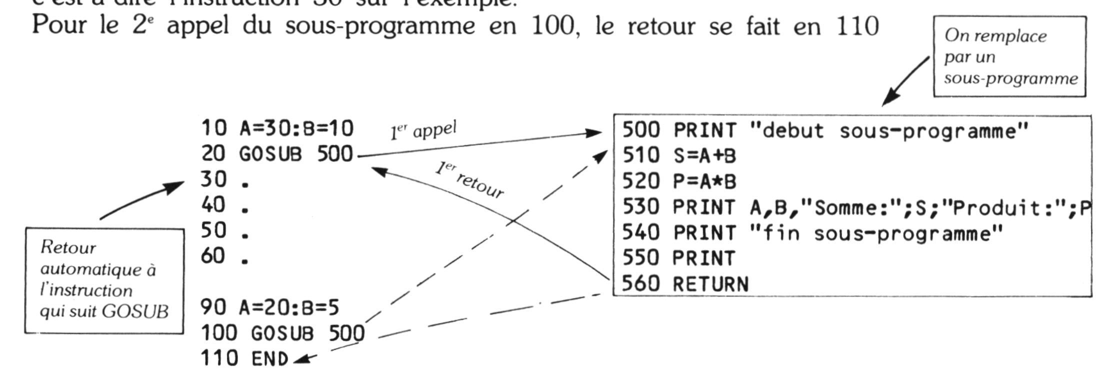
LES SOUS PROGRAMMES : GOSUB/RETURN

Il est fréquent qu'une même séquence d'instructions soit utilisée PLUSIEURS FOIS dans un programme.

Un sous programme permet d'écrire UNE SEULE FOIS cette séquence qu'il suffit d'appeler de différents endroits du programme par GOSUB no d'instruction.



20 GOSUB 500 provoque un branchement du programme en 500 (comme le ferait GOTO 500) mais **l'instruction RETURN** (RETOUR) **placée à la fin du sous-programme provoque un RETOUR AUTOMATIQUE après l'instruction qui suit GOSUB 500,** c'est-à-dire l'instruction 30 sur l'exemple.
Pour le 2e appel du sous-programme en 100, le retour se fait en 110



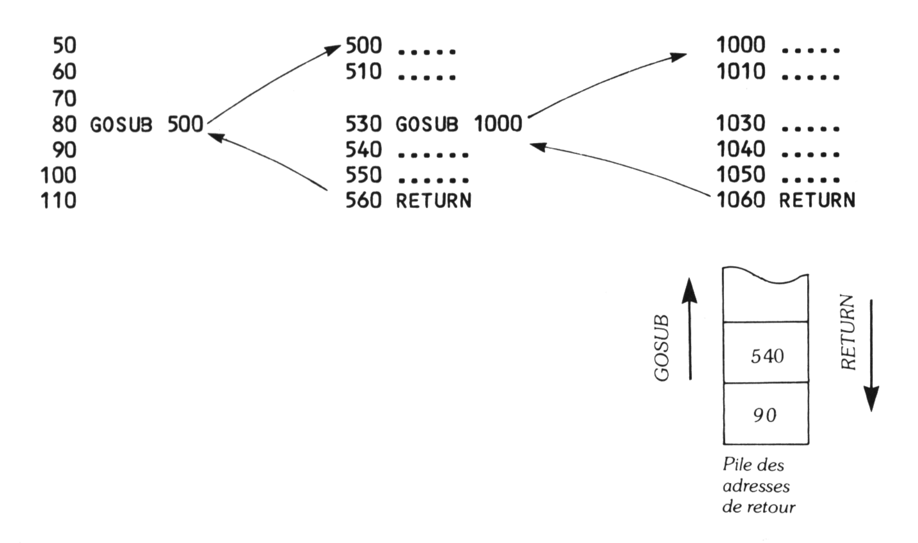
```
RUN

debut sous-programme
30  10  Somme:40  Produit:300
fin sous-programme

debut sous-programme
20  5   Somme:25  Produit:100
fin sous-programme
```

Un sous-programme peut lui-même en appeler un autre.

Les adresses de retour (90 et 540 sur l'exemple) sont gérées par BASIC à l'aide d'une pile.



Les instructions sont exécutées dans l'ordre suivant :

```
50 60 70 80
500 510 520 530              ' Debut du 1er sous programme
1000 1010 ....1050 1060      ' 2eme sous-programme
540 550 560                  ' Fin du 1er sous programme
90 100                       ' Retour au programme principal
```

N'essayez pas de sortir d'un sous-programme par GOTO, ni d'entrer dans un sous-programme par GOTO.

INTERLUDE

Au sujet des organigrammes

Faut-il utiliser des organigrammes ?

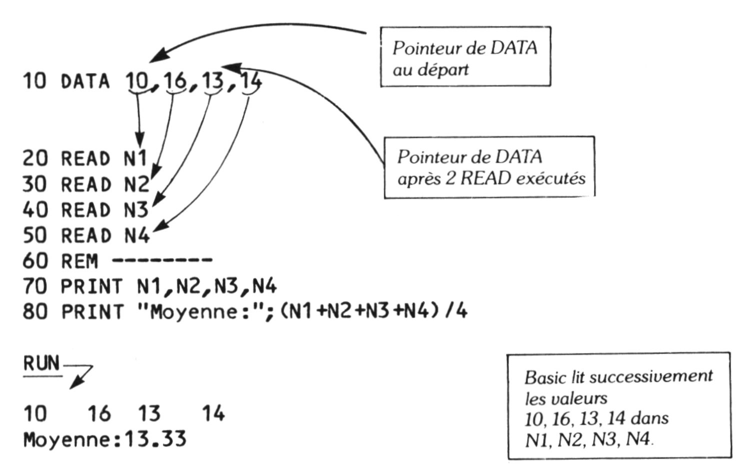
Ce sujet est très controversé ; certains en sont partisans, d'autres, au contraire ne les utilisent pas.

Les partisans des organigrammes prétendent qu'une programmation doit être précédée d'un organigramme, qu'un programme seul ne peut être directement compréhensible.

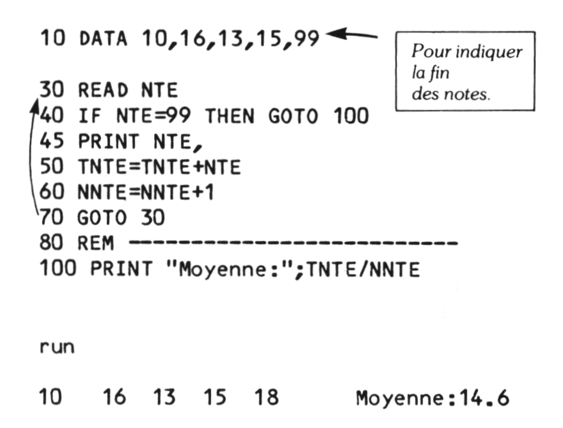
Ceux qui ne les utilisent pas prétendent qu'un programme bien structuré n'a pas besoin d'organigramme, qu'un organigramme trop développé devient vite confus.
Nous pensons qu'**il est important d'accorder beaucoup de soin à la présentation des programmes** de façon à ce qu'ils puissent être compris avec le minimum d'efforts.
Un organigramme 'suit' rarement un programme ; seul le programme reste. **C'est pourquoi, un programme doit être documenté le plus possible.**

DATA/READ/ RESTORE

L'instruction DATA permet de définir des données dans le programme lui-même. Celles-ci sont ensuite lues dans des variables par l'instruction 'READ variable' (LIRE variable).



'READ N1' lit la 1re donnée (10) dans N1.
Le pointeur de DATA (géré par BASIC) progresse de 1.
Ainsi 'READ N2' lit la 2e donnée dans N2, etc...
Sur l'exemple ci-dessus, nous avons supposé que le nombre de données à lire (4) était connu.
Pour repérer la fin des données, plaçons 99.



Les données peuvent être écrites sur plusieurs lignes :

```
10 DATA 10,16,13
20 DATA 15,18,99
```

est équivalent à la ligne 10 du programme ci-dessus.

Caractères spéciaux ;

Les DATAS contenant des caractères spéciaux doivent être placés entre guillemets. Sans la présence de guillemets, la virgule serait considérée comme séparateur.

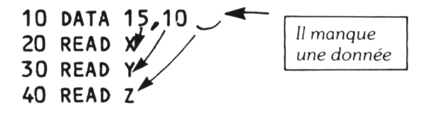
```
10 DATA "12,Rue LAGAFFE"
20 REM
30 READ X$
40 PRINT X$

RUN

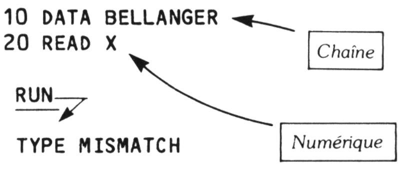
12,Rue LAGAFFE
```

Erreurs :

OUT OF DATA : Si le nombre de READ exécutés est supérieur au nombre de données en DATA, le message OUT OF DATA apparaît.



TYPE MISMATCH : Le type de la donnée lue doit s'accorder avec le type de la variable.



La donnée (type chaîne) ne s'accorde pas avec le type de la variable (type numérique)

Il faut écrire : 20 READ X$

RESTORE :

Positionne en début de DATA, ce qui permet de relire les données depuis le début.

```
 10 DATA 6,3,14
 20 :
 30 READ A
 40 READ B
 50 READ C
 60 :
 70 RESTORE
 80 :
 90 READ D,E,F
100 :
110 PRINT A,B,C
120 PRINT D,E,F

RUN

6  3  14
6  3  14
```

Annuaire téléphonique :

Un annuaire téléphonique est défini en DATAS. En fournissant un nom, on obtient le numéro de téléphone correspondant.

```
10 REM ANU                      ANNUAIRE
15 :
30 DATA BISSON,    444-22-11
40 DATA BELLANGER,555-33-22
45 DATA CAMPAS,    777-44-11
50 DATA *
60 REM-----------------------------------------------------
70 PRINT
80 INPUT "Quel nom? ";NOM$
85 :
90 RESTORE                       :REM Positionnement en debut
100 :
110 READ N$:IF N$="*" THEN PRINT "N'existe pas":GOTO 70
120 READ TPH$
130 IF N$=NOM$ THEN PRINT:PRINT N$,TPH$:GOTO 70
140 GOTO 110
150 '
160 ' RUN
170 '
180 ' Quel nom? CAMPAS
190 '
200 ' CAMPAS   777-44-11
```

Pour éviter d'entrer le nom en entier, lors d'une recherche, changer la ligne 130 :

```
130 IF NOM$=LEFT$(N$,LEN(NOM$)) THEN PRINT N$,TPH$:GOTO 70

RUN

Quel nom? BIS

BISSON     444-33-22
```

Naturellement, si plusieurs noms commencent par 'BIS', nous obtenons le premier de la liste.

La liste triée de l'annuaire s'obtiendrait en lisant les DATAS dans 2 tables qu'il faudrait ensuite trier.

Pluriel des noms se terminant par 'OU'

Nous savons que le pluriel des noms se terminant par 'OU' s'obtient en ajoutant 'S' au nom au singulier sauf pour CHOU, GENOU,...

```
10 REM PLUR          PLURIEL DE NOMS SE TERMINANT PAR 'OU'
30 DATA CHOU,GENOU,HIBOU,CAILLOU,BIJOU,JOUJOU,POU
40 :
50 PRINT
60 INPUT "Donnez un nom se terminant par 'OU' :";N$
70 IF RIGHT$(N$,2)<>"OU" THEN PRINT:PRINT "PAR OU ! ":GOTO 50
80 :
90 RESTORE
100 :
110 FOR L=1 TO 7
120  READ X$
130  IF X$=N$ THEN TERM$="X":GOTO 170
140 NEXT L
150 :
160 TERM$="S"
170 PRINT "Le pluriel est:";N$+TERM$
180 GOTO 50

run

Donnez moi un nom se terminant par 'OU' : VERROU
Le pluriel est: VERROUS

Donnez moi un nom se terminant par 'OU' : BIJOU
Le pluriel est: BIJOUX
```

Budget familial :

Dans chaque ligne de DATAS sont définies, dans l'ordre, les dépenses d'un mois : loyer, alimentation, divers. Le programme lit les DATAS et édite le total des dépenses pour chaque mois ainsi que le cumul par type de dépense.

```
10 REM- BUDG             BUDGET FAMILIAL
20 :                 Loyer  Alimentation  Divers
25 :                   ↓        ↓          ↓
40 DATA JANVIER, 2500,   3000,   1000
50 DATA FEVRIER, 2500,   2000,   1500
60 DATA MARS   , 2500,   2200,    600
70 DATA *
80 REM -------------------------------------------------
90 TLOYER=0:TALIM=0:TDIV=0
100 :
110 LPRINT ,"LOYER","ALIM","DIVERS"
120 :
130 FOR MOIS=1 TO 12
140  READ MOIS$:IF MOIS$="*" THEN 230
150  READ LOYER
160  READ ALIM
170  READ DIV
180  TMOIS=LOYER+ALIM+DIV
190  TLOYER=TLOYER+LOYER:TALIM=TALIM+ALIM:TDIV=TDIV+DIV
200  LPRINT MOIS$,LOYER,ALIM,DIV,TMOIS
210 NEXT MOIS
220 :
230 LPRINT
240 LPRINT ,TLOYER,TALIM,TDIV  <-- LPRINT affiche sur l'imprimante
```

```
              LOYER         ALIM          DIVERS
JANVIER        2500          3000          1000          6500
FEVRIER        2500          2000          1500          6000
MARS           2500          2200          600           5300

               7500          7200          3100
```

Devinez un mot :

Ce programme choisi un mot parmi ceux définis en DATA
Le jeu consiste à deviner ce mot.
Le joueur propose des lettres et le programme lui indique en retour la ou les positions des lettres dans le mot cherché.

Amélioration possible :

Les mots sont proposés dans l'ordre où ils sont définis en DATA.
Ils pourraient être choisis au hasard en faisant :

```
25 N=5                               :REM N:nombre de mots
80 RESTORE
82 X=RND(1)*N
84 FOR I=1 TO X:READ MOT$:NEXT I     :REM lecture de X mots
86 READ MOT$
```

MOT$ | M | A | I | S | O | N |

BPL( ) | O | O | O | | | |

0/1 0/1 — 1 si trouvé

```
10 REM DEVINE
30 DATA MAISON,SAPIN,BROUETTE,CIDRE,CASSETTE
40 DATA *
50 :
60 DIM BPL(15)                  :REM Table des lettres trouvees
70 :
80 READ MOT$:IF MOT$="*" THEN RESTORE:GOTO 80
90 L=LEN(MOT$)
100 FOR P=1 TO 15:BPL(P)=0:NEXT P
110 PRINT:PRINT "Vous devez trouver un mot de ";L;"Lettres":PRINT
120 :
130 PRINT:INPUT "Donnez une lettre:";L$
140 :
150 FOR P=1 TO L                :REM Recherche si lettre dans mot
160  IF L$=MID$(MOT$,P,1) THEN BPL(P)=1
170 NEXT P
180 REM --------------------------------------- Affichage resultat
190 BPL=0
200 FOR P=1 TO L
210  IF BPL(P)=1 THEN PRINT MID$(MOT$,P,1);:BPL=BPL+1:GOTO 230
220  PRINT ".";
230 NEXT P
240 IF BPL<L THEN 130
250 PRINT:PRINT "C'est bon"
260 GOTO 70
270 '
280 ' RUN ↵
290 '
300 ' Vous devez trouver un mot de 6 lettres
310 '
320 ' Donnez une lettre: S ↵
330 '
340 ' ...S..
350 '
360 ' Donnez une lettre:M ↵
370 '
380 ' M..S..
```

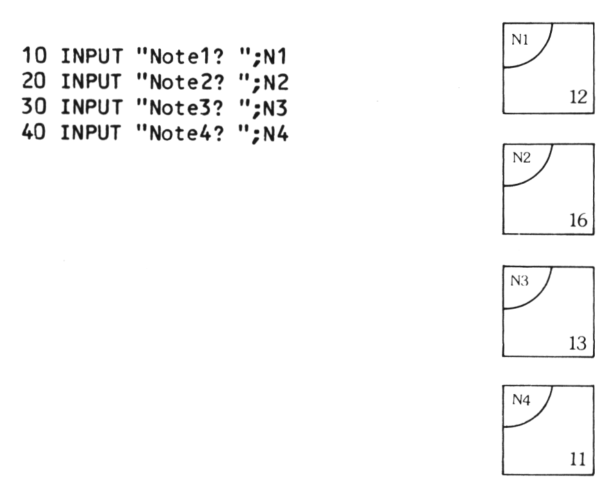
Questions/Réponses

Ce programme pose une question, propose des solutions puis vérifie si la réponse est bonne.

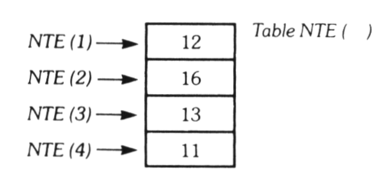
```
10 REM QRS              QUESTIONS/REPONSES  (avec solutions proposees)
20 :                                                              Bonne
30 :              Question                   Réponses proposées    réponse
50 DATA "Capitale de l'ESPAGNE? ",  "MADRID BRUXELLES MILAN",  "MADRID"
60 DATA "Date de MARIGNAN?      ",  "1515 1958 ",              "1515"
70 DATA *
80 :
90 READ Q$:IF Q$="*" THEN STOP          :REM lecture d'une question
100 READ RP$                            :REM reponses proposees
110 READ BR$                            :REM bonne reponse
120 :
130 PRINT:PRINT Q$;RP$                  :REM Question/reponses possibles
140 :
150 INPUT "Votre Reponse? ";REP$:IF REP$="" THEN 150
160 :
170 IF REP$=BR$ THEN PRINT:PRINT "C'est bon":GOTO 90
180 PRINT:PRINT "La reponse est:";BR$
190 GOTO 90
200 '
210 ' RUN
220 '
230 ' Capitale de l'ESPAGNE? MADRID BRUXELLES MILAN?
240 ' Votre reponse? MADRID
250 '
260 ' C'est bon
```

LES TABLES

Pour mémoriser 4 notes, nous pourrions utiliser 4 variables N1, N2, N3, N4 en faisant :



Utilisons plutôt une 'TABLE' que nous appelons NTE().
Les 'éléments' de cette table NTE() sont connus sous les noms de NTE(1)
NTE(2)
NTE(3)
NTE(4)



Nous pourrions faire comme ci-dessus :

```
10 INPUT "Note 1";NTE(1)
20 INPUT "Note2? ";NTE(2)
30 INPUT "Note3? ";NTE(3)
40 INPUT "Note4? ";NTE(4)
```

Mais utilisons plutôt une 'boucle FOR' ; en faisant varier un 'indice', nous simplifions l'écriture du programme :

```
        10 FOR N=1 TO 4
4 fois  20  INPUT "Note? ";NTE(N)
        30 NEXT N
```

La 1ère fois : est équivalent à NTE (1) puisque N = 1

Au départ, N est égal à 1. Par conséquent

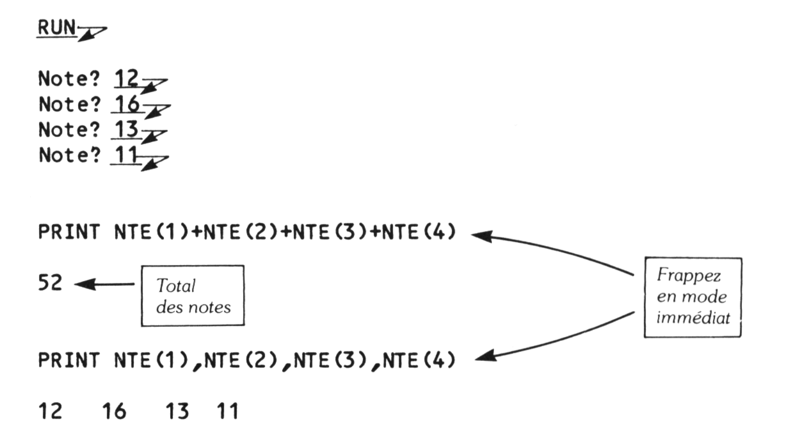
```
20 INPUT "Note? ";NTE(N) est equivalent a

20 INPUT "Note? ";NTE(1)
```

C'est donc dans NTE(1) que la première note est introduite.

```
              ------
12 ==>       ! 12 ! NTE(1)
             ! 0  ! NTE(2)
             ! 0  ! NTE(3)
             ! 0  !
              ------
```

Au second passage dans la boucle FOR, NTE(N) est équivalent à NTE(2) puisque N est devenu égal à 2.



Sans la boucle FOR, nous ferions :

```
 10 N=1
 20 :
 30 INPUT "Note? ";NTE(N)
 40 N=N+1
 50 IF N>5 THEN 100
 60 GOTO 30
 70 :

100 STOP
```

Edition de la table NTE() :

Editons la table NTE() après l'avoir documentée.

```
10 FOR N=1 TO 4
20  INPUT "Note? ";NTE(N)
30 NEXT N
40 REM ----------- Edition de la table NTE()
50 PRINT
60 PRINT "Liste des notes"
70 PRINT
80 FOR N=1 TO 4
90  PRINT "Note";N;NTE(N)
100 NEXT N

RUN

Note? 12
Note? 16
Note? 13
Note? 11

Liste des notes

Note 1 12
Note 2 16
Note 3 13
Note 4 11
```

Dimensionnement des tables : DIM (nombre d'éléments)

Les tables de plus de 10 éléments doivent être 'dimensionnées', afin de réserver leur place en mémoire centrale.

Les notes pourraient être définies dans une ligne DATA :

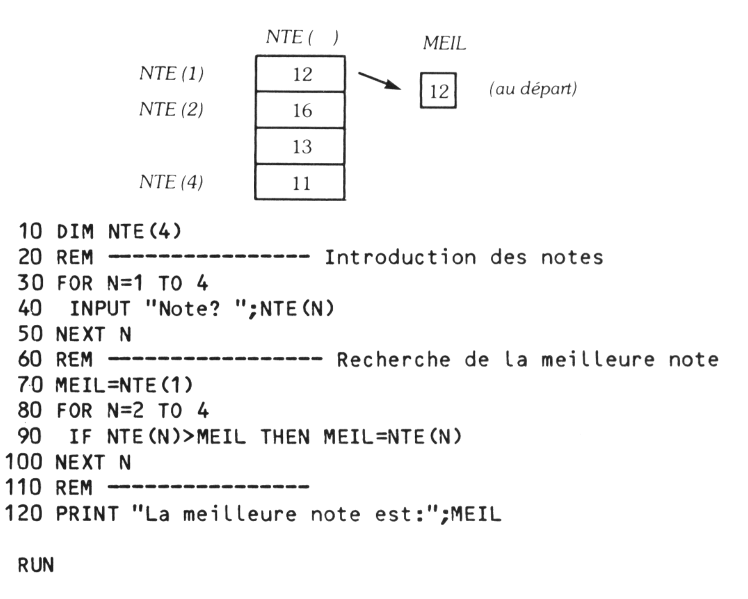
```
 10 DATA 12,16,13,11
 20 :
 25 DIM NTE(4)
 30 FOR N=1 TO 4       :REM lecture des DATAS
 40  READ NTE(N)
 50 NEXT N
 60 REM --------------- Total et moyenne
 70 TNTE=0
 90 FOR N=1 TO 4
100  TNTE=TNTE+NTE(N)
110 NEXT N
120 :
130 PRINT "Total/Moyenne:";TNTE,TNTE/4

  RUN

 Total/Moyenne:  52      13
```

Recherche de la meilleure note

Nous supposons d'abord que la première note est la meilleure. Puis nous la comparons à la seconde.
Si celle-ci est supérieure, elle devient la meilleure. Etc...

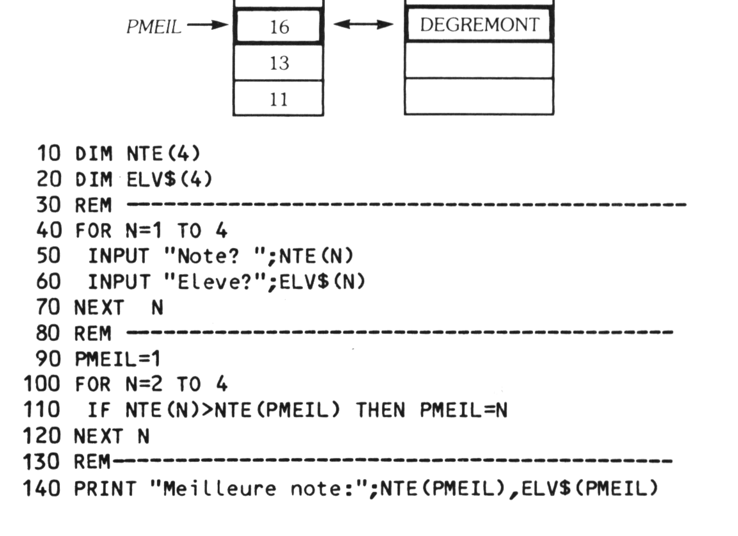



```
10 DIM NTE(4)
20 REM --------------- Introduction des notes
30 FOR N=1 TO 4
40  INPUT "Note? ";NTE(N)
50 NEXT N
60 REM ---------------- Recherche de la meilleure note
70 MEIL=NTE(1)
80 FOR N=2 TO 4
90  IF NTE(N)>MEIL THEN MEIL=NTE(N)
100 NEXT N
110 REM ---------------
120 PRINT "La meilleure note est:";MEIL

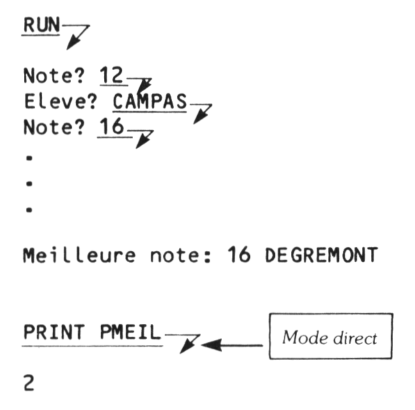
RUN

La meilleure note est: 16
```

Ci-dessous, nous ne mémorisons pas la meilleure note mais sa POSITION dans ta table.




```
10 DIM NTE(4)
20 DIM ELV$(4)
30 REM -------------------------------------------
40 FOR N=1 TO 4
50  INPUT "Note? ";NTE(N)
60  INPUT "Eleve?";ELV$(N)
70 NEXT  N
80 REM ------------------------------------------
90 PMEIL=1
100 FOR N=2 TO 4
110  IF NTE(N)>NTE(PMEIL) THEN PMEIL=N
120 NEXT N
130 REM------------------------------------------
140 PRINT "Meilleure note:";NTE(PMEIL),ELV$(PMEIL)
```



Remarque : Les 'traitements' que nous avons effectué (total, moyenne, recherche de la meilleure note) ne nécessitaient pas de table. Mais des traitements plus complexes tels que le tri par exemple rendent nécessaires l'utilisation de tables.

Tri des notes :

Trions les notes contenues dans la table NTE().
Nous comparons d'abord le 2e élément de la table au 1er :

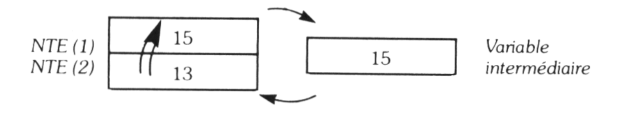
- S'il est plus grand, nous les laissons dans l'ordre.
- S'il est plus petit, nous les inversons afin qu'ils soient dans l'ordre.

```
230 IF NTE(I+1)<NTE(I) THEN X=NTE(I):NTE(I)=NTE(I+1):NTE(I+1)=X:IV=1

      !                   !

    SI NTE(I+!)<NTE(I)  ALORS   inversion de NTE(I+1) et de NTE (I)
```

L'inversion de NTE(I) et de NTE(I+1) se fait en utilisant une variable intermédiaire (X) :



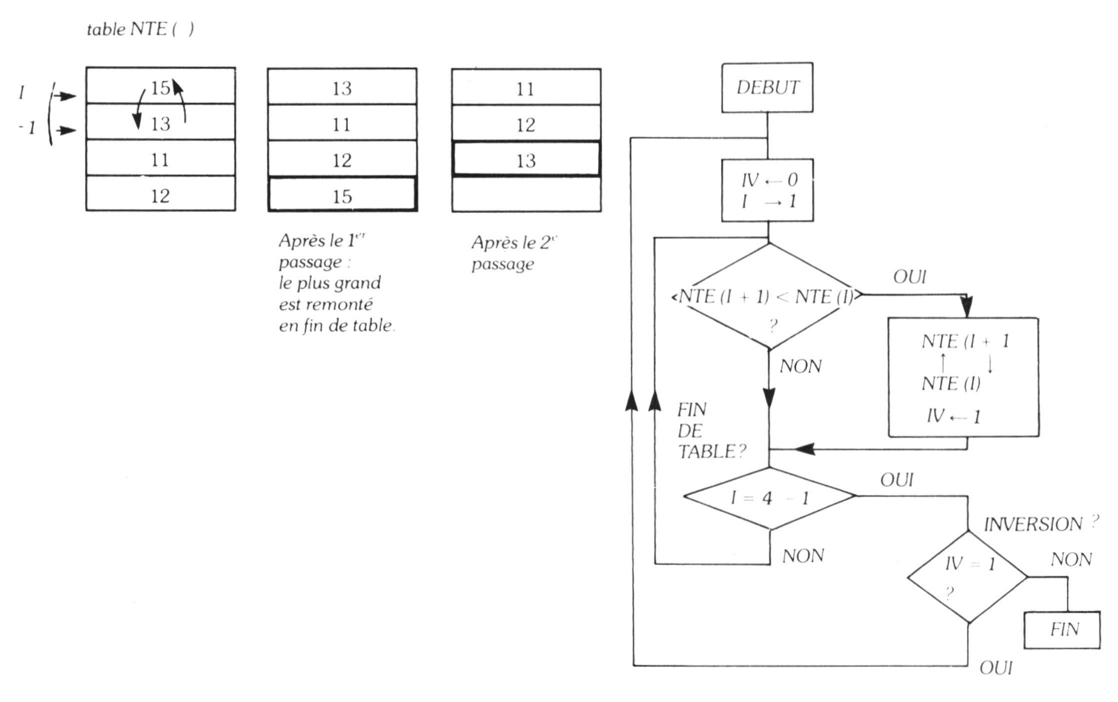
Ensuite, nous comparons de la même façon le troisième élément au second (en faisant progresser l'indice I de 1) et nous les inversons s'ils ne sont pas dans l'ordre. Etc...

Après avoir comparé tous les éléments adjacents de la table, le plus grand des éléments de la

table est 'monté' en fin de table. (Remarquez qu'il n'y a que N-1 comparaisons pour N éléments). Mais le tri n'est pas nécessairement achevé.

Pour le savoir, il suffit de tester le témoin d'inversion IV. S'il n'y a pas d'inversion au cours de l'exploration de la table, c'est que NTE(1) < NTE(2) < NTE(3) < NTE(4). Par conséquent, les éléments sont dans l'ordre.

En revanche, s'il y a eu inversion, nous explorons à nouveau la table. A l'issue de la seconde exploration, le plus grand des N-1 éléments est arrivé en avant-dernière position. N explorations au maximum sont nécessaires pour trier la table.




```
40 FOR I=1 TO 4
50  INPUT "Note? ";NTE(I)
70 NEXT I
200 REM------------------------------ Tri des notes -----------------
205 :
210 IV=0                            :REM Temoin d'inversion
220 FOR I=1 TO 4-1
230  IF NTE(I+1)<NTE(I) THEN X=NTE(I):NTE(I)=NTE(I+1):NTE(I+1)=X:IV=1
240 NEXT I
250 IF IV=1 THEN 210                :REM Y a t'il eu inversion?
255 :
260 REM-------------------- Edition des notes triees ----------------
270 FOR I=1 TO 4
280   PRINT NTE(I)
290 NEXT I
```

```
RUN

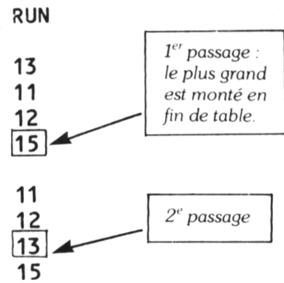
Note? 15
Note? 13
Note? 11
Note? 12

11
12
13
15
```

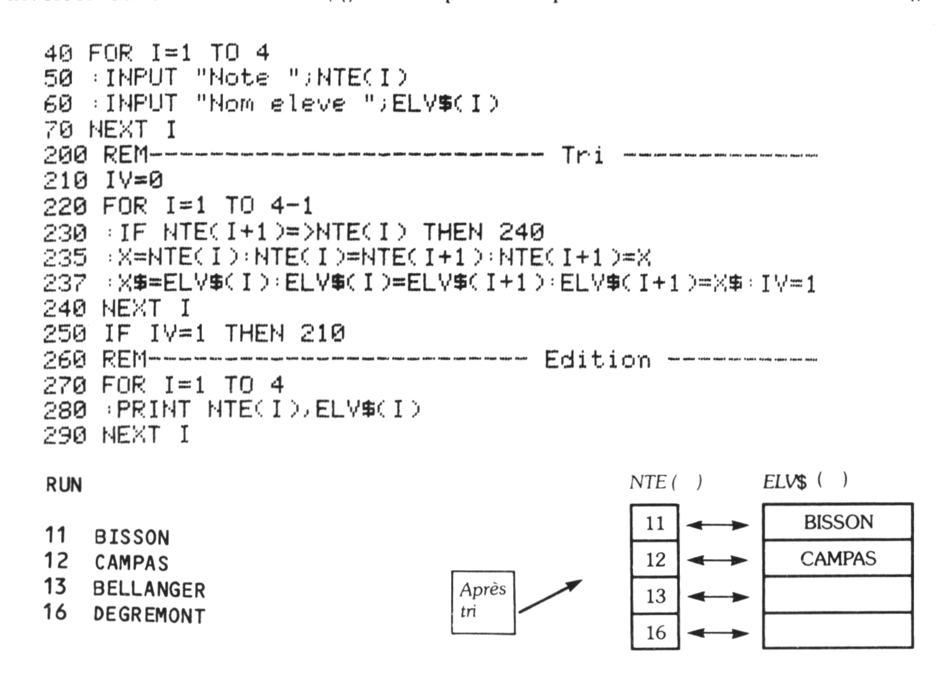
Pour mieux 'suivre' l'évolution du tri, vous pouvez insérer :

```
245 FOR K=1 TO 4:PRINT NTE(K):NEXT K:PRINT
```

Après chaque exploration de table, nous éditons son contenu.



Pour obtenir la liste des notes dans l'ordre croissant accompagnée de la liste des noms, il suffit d'inverser dans la table ELV$() à chaque fois que nous inversons dans NTE().



```
40 FOR I=1 TO 4
50 :INPUT "Note ";NTE(I)
60 :INPUT "Nom eleve ";ELV$(I)
70 NEXT I
200 REM------------------------- Tri -------------
210 IV=0
220 FOR I=1 TO 4-1
230 :IF NTE(I+1)=>NTE(I) THEN 240
235 :X=NTE(I):NTE(I)=NTE(I+1):NTE(I+1)=X
237 :X$=ELV$(I):ELV$(I)=ELV$(I+1):ELV$(I+1)=X$:IV=1
240 NEXT I
250 IF IV=1 THEN 210
260 REM------------------------ Edition ----------
270 FOR I=1 TO 4
280 :PRINT NTE(I),ELV$(I)
290 NEXT I
```

```
RUN

11  BISSON
12  CAMPAS
13  BELLANGER
16  DEGREMONT
```

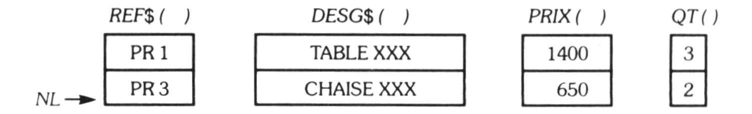
Pour obtenir la liste triée dans l'ordre croissant des noms, il suffit de faire :

```
230 IF ELV$(I+1)<ELV$(I) THEN ........
```

Etablissement d'une facture :

En DATAS, sont définis des produits (référence, désignation et prix)

Les produits facturés sont stockés dans 4 tables REF$(), DESG$(), PRIX() et QT(). Lorsque tous les produits sont saisis, nous éditons la facture.



Exécution :

```
Nom client? DUPONT
Adresse?   15 rue xxxxxxxxxxxx

Produit (F pour fin)? PR1
Quantite?             3
Produit (F pour fin)? PR3
Quantite?             2
Produit (F pour fin)? F


Etablissements xxx

                        Facture de: DUPONT xxxxx
                                    15 rue xxxxxxxxx

PR1      TABLE xxxxxx              1400   3  4200
PR3      CHAISE xxxxx              650    2  1300


                              Total:         5500
```

```
10 REM FT       FACTURE
20 :
30 REM---------- References,designations,Prix des Produits ---
40 :
50 DATA "PR1","FAUTEUIL XXXXX",1400
60 DATA "PR2","TABLE XXXXXXXX",2100
70 DATA "PR3","CHAISE XXXXXXX",650
80 DATA *
90 REM------------------------- Saisie des references----------
100 INPUT "Nom client ";NOM$
110 INPUT "Adresse    ";ADR$
120 :
130 NL=0                 :REM NL= nb de lignes
140 :
150 INPUT "Reference Produit (F Pour fin) ";R$
160 IF R$="F" THEN 330
170 INPUT "Quantite ";QT
180 :
190 RESTORE                    :REM Positionnement debut DATA
200 :
210 READ REF$:IF REF$="*" THEN PRINT "N'existe Pas":GOTO 150
220 READ DESG$
230 READ PRIX
240 IF R$<>REF$ THEN 210
250 NL=NL+1
260 REF$(NL)=REF$
270 DESG$(NL)=DESG$
280 PRIX(NL)=PRIX
290 QT(NL)=QT
300 :
310 GOTO 150
320 REM--------------------------------Edition facture-----------
330 LPRINT
340 LPRINT "Etablissement xxx":LPRINT
350 LPRINT TAB(13+28);"Facture de:";TAB(13+40);NOM$
360 LPRINT TAB(13+40);ADR$
370 LPRINT
380 T=0
390 FOR L=1 TO NL
400 :LPRINT REF$(L);
410 :LPRINT TAB(13+6);DESG$(L);
420 :LPRINT TAB(13+30);PRIX(L);
430 :LPRINT TAB(13+40);QT(L);
440 :LPRINT TAB(13+50);QT(L)*PRIX(L)
450 :T=T+QT(L)*PRIX(L)
460 NEXT L
470 LPRINT
480 LPRINT TAB(13+40);"Total:";TAB(13+50);T
```

Composez votre menu :

Vous avez le choix entre 3 hors-d'œuvres, 3 plats et 3 desserts. Vous ne disposez que de 60 f.

Toutes les combinaisons de hors-d'œuvres, de plats et de desserts sont calculées. Seules sont retenues celles dont les montants sont inférieurs ou égaux à 60.
Pour afficher les noms des hors-d'œuvres, plats et desserts, il faudrait les définir en DATA et les lire dans 3 tables (comme pour les prix).

```
10 REM RESTA
20 :
30 BUDG=60                                :REM Somme disponible
40 :
50 DATA 8,10,12
60 DATA 35,40,45
70 DATA 12,15,20
80 :
90 FOR H=1 TO 3:READ PH(H):NEXT H
100 FOR P=1 TO 3:READ PP(P):NEXT P
110 FOR D=1 TO 3:READ PD(D):NEXT D
120 :
130 FOR H=1 TO 3
140  FOR P=1 TO 3
150    FOR D=1 TO 3
160      T=PH(H)+PP(P)+PD(D)
170      IF T>BUDG THEN 190
180      LPRINT "Hors d'oeuvre:";H;"Plat:";P;"Dessert:";D;" Total:";T
190   NEXT D
200  NEXT P
210 NEXT H
```

PH( )	*PP( )*	*PD( )*
8	35	12
10	40	45
12	15	20
Prix hors d'œuvres	*Prix plats*	*Prix desserts*

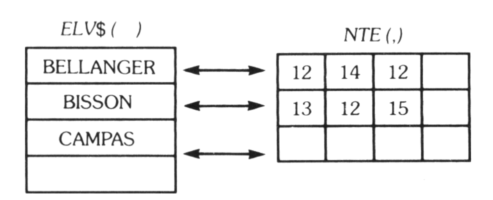
```
Hors d'oeuvre: 1 Plat: 1 Dessert: 1  Total: 55
Hors d'oeuvre: 1 Plat: 1 Dessert: 2  Total: 58
Hors d'oeuvre: 1 Plat: 2 Dessert: 1  Total: 60
Hors d'oeuvre: 2 Plat: 1 Dessert: 1  Total: 57
Hors d'oeuvre: 2 Plat: 1 Dessert: 2  Total: 60
Hors d'oeuvre: 3 Plat: 1 Dessert: 1  Total: 59
```

Table à 2 dimensions :

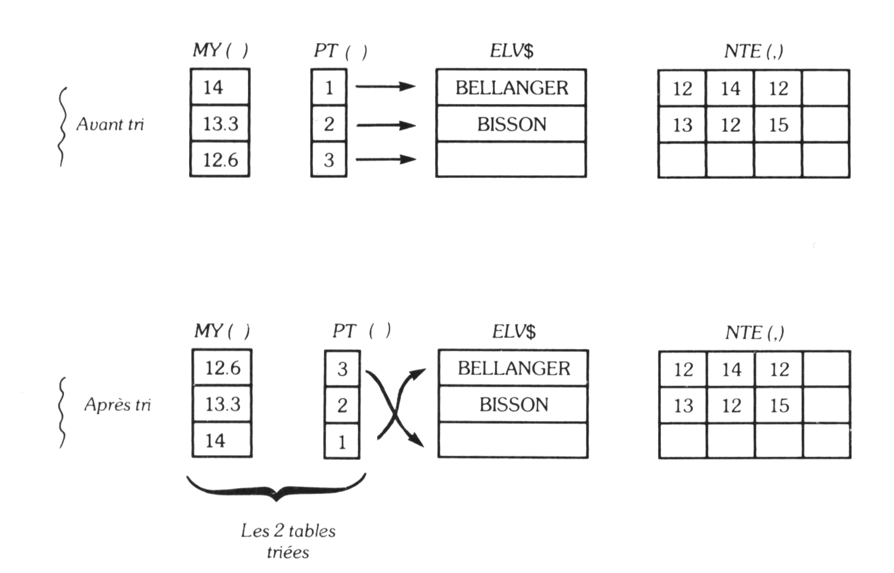
Les tables peuvent avoir plusieurs dimensions (jusqu'à 255).

Soit un ensemble de notes définies en DATA. Nous voulons les éditer. Nous pourrions lire les DATAS et les éditer au fur et à mesure (voir chapitre sur les DATA).

L'utilisation d'une table à 2 dimensions permet d'effectuer d'autres traitements (comme le tri par exemple). Chaque ligne représente les notes d'un élève.



Pour obtenir la liste des élèves et de leurs notes dans l'ordre décroissant des moyennes, nous utilisons une table intermédiaire (PT).
Cette table PT() 'pointe' vers la table des notes NTE(,). Le tri se fait sur la table MY() (en inversant les éléments de PT() à chaque fois que nous inversons 2 éléments de MY()). Sans cette table PT(), il faudrait inverser les notes de la table NTE(,), ce qui serait fastidieux.



```
10 REM TN      ANALYSE DE NOTES
20 :
30 DATA BELLANGER,   12,14,12,99
40 DATA BISSON,      13,12,15,99
50 DATA CAMPAS,      16,10,16,99
60 DATA *
70 REM------------------------------------------ Lecture des DATAS ----
80 DIM ELV$(30),MY(30),PT(30)
90 DIM NTE(30,10)
100 :
110 FOR E=1 TO 30           30 eleves maxi
120 :READ X$:IF X$="*" THEN NE=E-1:GOTO 210
130 :ELV$(E)=X$
140 :
150 :FOR N=1 TO 10          10 notes max par eleve
160 : READ X:IF X=99 THEN 190
170 : NTE(E,N)=X
180 :NEXT N
190 NEXT E
200 REM---------------------------- Editions de ELV$( ) et NTE( , ) --
210 FOR E=1 TO NE
220 :LPRINT ELV$(E);TAB(25);
230 :FOR N=1 TO 10
240 : IF NTE(E,N)=0 THEN 270
250 : LPRINT TAB(25+N*3);NTE(E,N);
260 :NEXT N
270 :LPRINT
280 NEXT E
290 REM------------------- Calcul des moyennes dans MY( ) ----
300 LPRINT:LPRINT "Moyennes":LPRINT
310 FOR E=1 TO NE
320 :TN=0
330 :FOR N=1 TO 10
340 : IF NTE(E,N)=0 THEN NN=N-1:GOTO 370
350 : TN=TN+NTE(E,N)
360 :NEXT N
370 :MY(E)=TN/NN
380 :LPRINT "Moy:";ELV$(E);TAB(30);MY(E)
390 NEXT E
400 REM================= Edition dans l'ordre des moyennes ====
410 FOR I=1 TO 30:PT(I)=I:NEXT I
420 REM---------------- Tri
430 IV=0
440 FOR I=1 TO NE-1
450 :IF MY(I+1)<=MY(I) THEN 460
455 :X=MY(I):MY(I)=MY(I+1):MY(I+1)=X
457 :X=PT(I):PT(I)=PT(I+1):PT(I+1)=X:IV=1
460 NEXT I
```

```
470 IF IV=1 THEN 430
480 REM------------------------- Edition
485 LPRINT:LPRINT "Liste dans l'ordre des moyennes ":LPRINT
490 FOR E=1 TO NE
500 :LPRINT ELV$(PT(E));TAB(25);
510 :FOR N=1 TO 10
520 : IF NTE(PT(E),N)=0 THEN 540
530 : LPRINT TAB(25+3*N);NTE(PT(E),N);
540 :NEXT N
550 :LPRINT TAB(40);MY(E)
560 NEXT E
```

```
BELLANGER       12 14 12
BISSON          13 12 15
CAMPAS          16 10 16

Moyennes

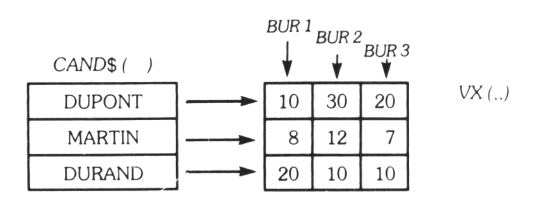
Moy:BELLANGER     12.6666667
Moy:BISSON        13.3333333
Moy:CAMPAS        14

Liste dans l'ordre des moyennes

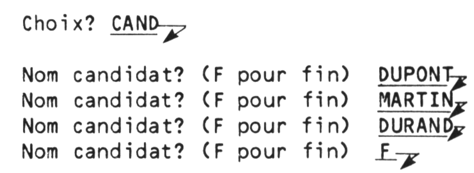
CAMPAS          16 10 16    14
BISSON          13 12 15    13.3333333
BELLANGER       12 14 12    12.6666667
```

Elections

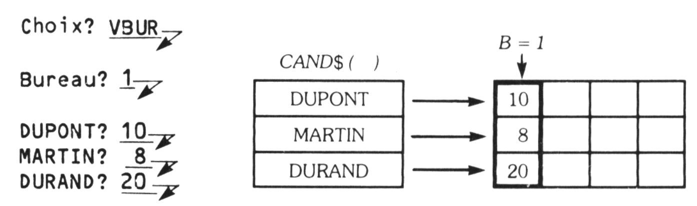
Plusieurs bureaux de vote sont gérés. Les résultats pour chaque candidat et chaque bureau de vote sont stockés dans une table VX(,) à 2 dimensions.



Un premier mode (CAND) permet de définir les noms des candidats et de documenter la table CAND$().



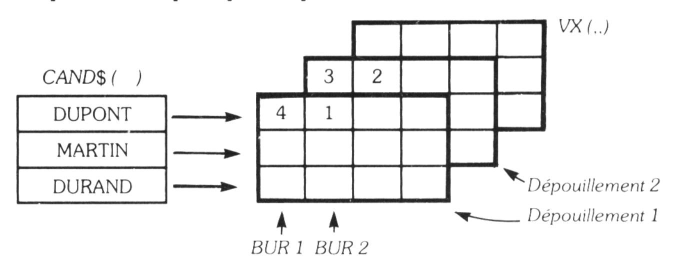
Un second mode (VBUR) permet de documenter la table VX(,) pour un bureau de vote (1, 2,...). Sur l'exemple, il n'y a pas cumul au fur et à mesure du dépouillement ; il faut entrer le nombre de voix total pour un bureau de vote. L'ancienne valeur peut cependant être modifiée.



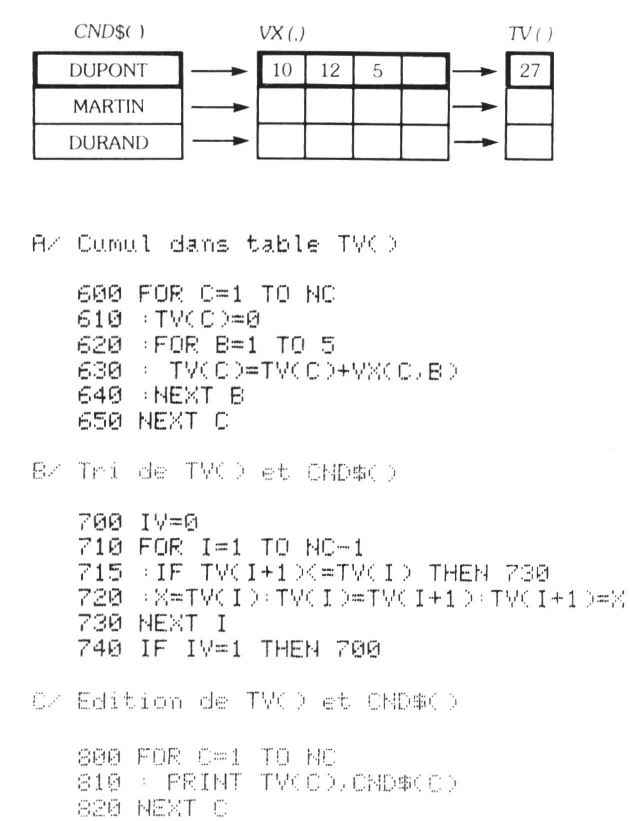
Un troisième mode affiche les résultats (RESUL) : total des voix pour chaque candidat ainsi que le pourcentage.

Pour effectuer le cumul par bureau de vote, au fur et à mesure du dépouillement, 2 solutions sont possibles :

- Avec la même table à 2 dimensions VX(,), ajouter les voix au fur et à mesure.
- Définir la table VX avec 3 dimensions et y consigner tous les dépouillements intermédiaires. Cette solution est plus 'sûre' puisqu'elle permet d'avoir la trace de tous les dépouillements.



Pour obtenir la liste des résultats dans l'ordre des voix obtenues par les candidats, nous pourrions créer une table TV() et la trier.



A/ Cumul dans table TV()

```
600 FOR C=1 TO NC
610 :TV(C)=0
620 :FOR B=1 TO 5
630 :  TV(C)=TV(C)+VX(C,B)
640 :NEXT B
650 NEXT C
```

B/ Tri de TV() et CND$()

```
700 IV=0
710 FOR I=1 TO NC-1
715 :IF TV(I+1)<=TV(I) THEN 730
720 :X=TV(I):TV(I)=TV(I+1):TV(I+1)=X
730 NEXT I
740 IF IV=1 THEN 700
```

C/ Edition de TV() et CND$()

```
800 FOR C=1 TO NC
810 :  PRINT TV(C),CND$(C)
820 NEXT C
```

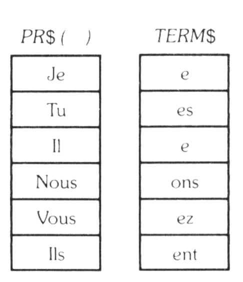
```
60  DUPOND
44  DURAND
27  MARTIN
```

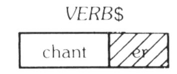
```
10 REM EL     ELECTIONS
20 :
50 DIM CND$(10),VX(10,5)     :REM 10 candidats/5 bureaux de vote
60 REM--------------------------- Choix -----------------------
75 CLS
80 PRINT:INPUT "Votre choix (CAND,VBUR,RESUL) ";CH$
90 IF CH$="CAND" THEN GOSUB 140
100 IF CH$="VBUR" THEN GOSUB 240
110 IF CH$="RESUL" THEN GOSUB 350
120 GOTO 80
130 REM-------------------- Noms des candidats -------------------
140 PRINT:INPUT "Nom candidat (F pour fin) ";C$:IF C$="F" THEN 220
160 FOR C=1 TO 10
170 :IF C$=CND$(C) THEN PRINT:PRINT "Existe deja":GOTO 140
180 :IF CND$(C)="" THEN CND$(C)=C$:NC=C:GOTO 140
190 NEXT C
200 PRINT "Trop de candidats":STOP
220 RETURN
230 REM--------------------------- Voix ---------------------------
240 PRINT:B=0:INPUT "Quel bureau (0 pour fin) ";B:IF B=0 THEN RETUR
N
250 :
260 FOR C=1 TO NC
270 :PRINT CND$(C);:X=10-LEN(CND$(C)):IF X>0 THEN PRINT SPC(X);
280 :PRINT VX(C,B);:PRINT SPC(4);
290 :V=0:INPUT "Voix (0 ou X) ";V:IF V=0 THEN 310
300 :VX(C,B)=V
310 NEXT C
320 RETURN
330 REM---------------------- Affichage resultats ------------
350 T=0
360 FOR C=1 TO NC
370 :FOR B=1 TO 5:T=T+VX(C,B):NEXT B
380 NEXT C
390 :
400 LPRINT:LPRINT "Resultats":LPRINT
410 FOR C=1 TO NC
420 :LPRINT CND$(C);TAB(13+15);
430 :TC=0
440 :FOR B=1 TO 5
450 : TC=TC+VX(C,B)
460 : IF VX(C,B)<>0 THEN LPRINT TAB(13+15+B*4);VX(C,B);
470 :NEXT B
480 :LPRINT TAB(13+40);TC;
490 :LPRINT TAB(13+45);"%";TC/T*100
500 NEXT C
510 RETURN
520 :
527 : Resultats
528 :
530 : DUPONT       10  30 20      60 %45.8
540 : MARTIN       8   12 7       27 %20.6
550 : DURAND       20  14 10      44 %33.5
```

Conjugaisons

Ce programme conjugue les verbes du 1er groupe au présent.

```
10 REM CONJ             CONJUGAISON
20 :
30 DATA JE,TU,IL,NOUS,VOUS,ILS
40 DATA E,ES,E,ONS,EZ,ENT
50 REM -------------------------------- Lecture des DATAS
60 FOR T=1 TO 6
70  READ PR$(T)
80 NEXT T
90 :
100 FOR T=1 TO 6
110  READ TERM$(T)
120 NEXT T
130 :
140 INPUT "Quel verbe? (1er groupe) ";VERB$
150 L=LEN(VERB$)
160 VERB$=LEFT$(VERB$,L-2)
170 :
180 FOR T=1 TO 6
190  PRINT PR$(T);" ";
200  PRINT VERB$;TERM$(T)
210 NEXT T
220 PRINT
230 GOTO 140
240 '
250 ' run
260 '
280 ' Quel verbe? (1er groupe) CHANTER
290 '
300 ' JE CHANTE
310 ' TU CHANTES
320 ' IL CHANTE
```





LES ÉDITIONS :

Nous présentons sommairement les cas d'édition les plus courants.

PRINT variable

PRINT suivi d'une variable ou d'une constante provoque l'édition de celle-ci puis d'un saut de ligne et d'un retour en début de ligne.
PRINT seul provoque un saut de ligne

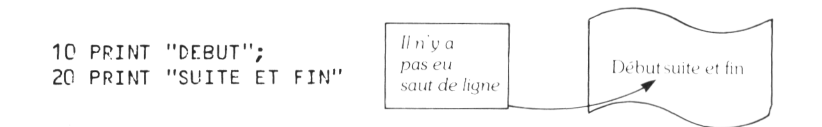
```
10 PRINT "DEBUT"
20 PRINT
30 PRINT "FIN"
```



PRINT variable ;

Le ; (point virgule) empêche le saut de ligne.

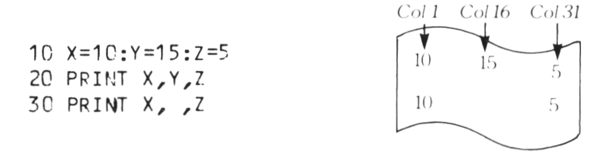
```
10 PRINT "DEBUT";
20 PRINT "SUITE ET FIN"
```



PRINT variable ,

La , (virgule) permet de cadrer les valeurs imprimées de façon standard en colonnes 1, 16, 31,...

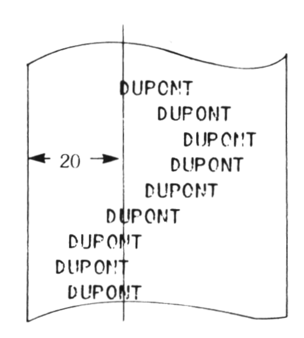
```
10 X=10:Y=15:Z=5
20 PRINT X,Y,Z
30 PRINT X, ,Z
```



PRINT TAB (position)

Place la tête d'impression (ou le curseur) dans la colonne spécifiée de la ligne courante, permettant ainsi les cadrages.

```
10 INPUT "Votre nom? ";NOM$
20 :
30 FOR A=0 TO 6.28*3 STEP 6.28/20
40  X=20+SIN(A)*10
50  PRINT TAB(X);NOM$
60 NEXT A
```



Dessin d'un triangle plein :

```
10 FOR L=1 TO 9 STEP 2
20  PRINT TAB(20-L/2);
30  FOR E=1 TO L
40    PRINT "*";
50  NEXT E
60  PRINT
70 NEXT L

RUN
```



ATTENTION :

• **L PRINT TAB ( )** ne fonctionne pas correctement (pour l'instant).

```
LPRINT TAB(13+5);"XXX"
```

affiche xxx en colonne 5. Par conséquent, il faut ajouter 13 aux instructions L PRINT TAB ( ).

• **PRINT TAB ( )** ne fonctionne pas correctement.
Pour afficher "Jean" en colonne 20, il faut faire :

```
10 NOM$="DUPONT":PR$="JEAN""
20 PRINT NOM$;SPC(20-LEN(NOM$));PR$
```

• **Pour éviter la perte de caractères lors d'une édition,** faire :

```
10 CALL #ED01:LPRINT "XXXXXXXX":CALL #ECC7
```

• **Pour lister un programme,**
ajouter :

```
 2 CALL #ED01:LLIST 10-
10 PROGRAMME A LISTER
```

puis faire :

```
RUN
```

Histogrammes

```
10 REM  HGAT         HISTOGRAMME
20 :
30 REM  Imprime un histogramme des ventes de produits
40 REM  La taille des cases est proportionnelle au pourcentage
50 REM  des ventes
60 :
70 N=4                              :REM nombre de produits
80 DATA PROD1,25,PROD2,12,PROD3,10,PROD4,40
90 FOR I=1 TO N
100 :READ HIST$(I)
110 :READ HIST(I)
120 NEXT I
130 REM------------- Echelle ----
140 TT=0
150 FOR I=1 TO N
160 :TT=TT+HIST(I)
170 NEXT I
180 :
190 ECH=20/TT
200 REM------------------------ Edition ---
205 LPRINT
210 LPRINT "--------------------"
220 FOR I=1 TO N
230 :LPRINT "!";TAB(13+1);HIST(I);TAB(13+5);HIST$(I);TAB(13+12);
240 :LPRINT "%";INT((HIST(I)/TT*100)+.5);
245 :LPRINT TAB(13+17);"!"
250 :FOR K=1 TO HIST(I)*ECH-2
260 : IF HIST(I)*ECH-1 <1 THEN 280
270 : LPRINT "!                !"
280 :NEXT K
290 :LPRINT "--------------------"
300 NEXT I
```

HIST$( )		HIST( )
PROD 1	↔	25
PROD 2	↔	12
	↔	10
	↔	40

```
------------------
!25  PROD1  %29  !
!                !
!                !
!                !
------------------
!12  PROD2  %14  !
!                !
------------------
!10  PROD3  %11  !
!                !
------------------
!40  PROD4  %46  !
!                !
!                !
!                !
!                !
!                !
!                !
!                !
------------------
```

```
10 REM     HISTOGRAMME
20 :
30 REM     Repartition par classes d'age
40 :
50 DATA 43,H,34,F,23,H,56,F,36,F,57,H
60 DATA 21,F,32,H,55,F,34,H,35,H,27,F
70 DATA 999
80 :
90 READ AGE:IF AGE=999 THEN 190
100 READ SX$
110 IF AGE<20 OR AGE>60 THEN 90
120 :
130 CL=INT(AGE-20)/5+1
140 IF SX$="H" THEN H(CL)=H(CL)+1
150 IF SX$="F" THEN F(CL)=F(CL)+1
160 :
170 GOTO 90
180 REM-------- Echelle
190 MX=0
200 FOR CL=1 TO 8
210 :IF H(CL)+F(CL)>MX THEN MX=H(CL)+F(CL)
220 NEXT CL
230 ECH=10/MX
240 :
250 REM--------------------------Edition -----------
260 LPRINT
270 LPRINT "Age";TAB(13+12);"Ensemble";TAB(13+25);
275 LPRINT "Hommes";TAB(13+35);"Femmes":LPRINT
280 FOR CL=1 TO 8
290 :LPRINT 20+5*(CL-1);"-";
300 :LPRINT 20+5*CL-1;
310 :LPRINT H(CL)+F(CL);TAB(13+12);
320 :
330 :X=(H(CL)+F(CL))*ECH
340 :FOR I=1 TO X
350 :  IF X<1 THEN 370
360 :LPRINT "*";
370 :NEXT I
380 :
390 :LPRINT TAB(13+25);
400 :X=H(CL)*ECH
410 :FOR I=1 TO X
420 :  IF X<1 THEN 440
430 :  LPRINT "H";
440 :NEXT I
450 :
460 :LPRINT TAB(13+35);
470 :X=F(CL)*ECH
480 :FOR I=1 TO X
490 :  IF X<1 THEN 510
500 :  LPRINT "F";
510 :NEXT I
520 :LPRINT
530 NEXT CL
```

```
Age          Ensemble     Hommes    Femmes

 20 - 24  2 ******        HHH       FFF
 25 - 29  1 ***                     FFF
 30 - 34  3 **********    HHHHHH    FFF
 35 - 39  2 ******        HHH       FFF
 40 - 44  1 ***           HHH
 45 - 49  0
 50 - 54  0
 55 - 59  3 **********    HHH       FFFFFF
```

Systèmes de numération :

Pour représenter le nombre 234, nous pourrions 'dessiner' 234 bâtons. Mais il est plus simple de dire que nous avons :

4 unités
3 dizaines (3 paquets de 10 bâtons)
2 centaines (2 paquets de 100 bâtons)

```
   !!     !!!      !!!!

   2       3       4
   ↓       ↓       ↓
 2*100  + 3*10   + 4*1
```

10 représente la 'base'. Nous pouvons tout aussi bien utiliser une base 8 :

```
  !!!    !!!!!    !!

   3       5      2
   ↓       ↓      ↓
 3*64 +  5*8    + 2*1    --> 234 en decimal
```

D'une façon générale, un nombre N en base B s'exprime ainsi :

$$N=an*B^{n} +an-1*B^{n-1} +.....+ a1*B^{1} + a0*B^{0}$$

1101 en base 2 est égal à :

$$1*2^{3}+1*2^{2}+ 0* 2^{1} +1* 2^{0} \text{ ---> } 1*8+1*4+0*2+1 \text{ --> } 13$$

```
10 REM convd    Conversion base B --> Decimal
20 :
30 INPUT "Base? ";B
40 :
50 INPUT "Nombre? ";N
60 ND=0
70 :
80 FOR P=0 TO 6
90   Q=INT(N/10)         :REM Quotient
100  R=N-(Q*10)          :REM Reste: donne le coefficient de rang p
110 :
120  ND=ND+(B^P)*R
130  N=Q
140 NEXT P
150 :
160 PRINT ND
170 GOTO 50
```

N | 1 | 0 | **1** |

1 *Reste de la division par 10*

```
RUN

Base? 2

Nombre? 101

5

Nombre? 110

6
```

Conversion décimal en base B :

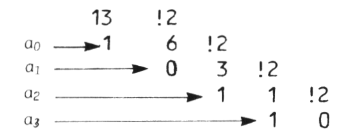
Exprimons un nombre N sous une autre forme.

$$N=an*B^{n}+an-1*B^{n-1}+....+a0*B^{0} = \underbrace{(an*B)+an-1)*B+...+a2)*B+a1)}_{Quotient}*B+\underbrace{a0}_{Reste}$$

En divisant N par B, nous obtenons un quotient Q = (an*B) + an − 1)*B +...+ a2)*B + a1) et un reste égal à a0.
En divisant le quotient par B, nous obtenons un reste égal à a1
Ainsi les restes des divisions successives par la base B donnent les coefficients a0, a1,..., an

Exemple :

Convertissons 13 (décimal) en base 2



1re solution :
Appliquons directement la méthode ci-dessus. Les coefficients étant obtenus dans l'ordre inverse de l'affichage, nous ne pouvons les éditer au fur et à mesure du calcul. Nous les stockons dans AF.

```
10 REM CONVB   Conversion Decimal --> Base B
20 :
30 INPUT "Base? ";B
40 :
50 INPUT "Nombre? ";N
60 :
70 AF=0:D=1                          :REM affichage resultat- Decalage
80 :
90 Q=INT(N/B)                        :REM Quotient
100 R=N-(Q*B)                        :REM Reste
110 :
120 AF=R*D+AF
130 D=D*10
140 :
150 N=Q
160 IF Q>0 THEN 90
170 :
180 PRINT AF
190 GOTO 50

RUN

Base? 2

Nombre? 5

101

Nombre? 6
```

	1	AF
1	0	R * D
1	1	AF

2e solution :
Nous commençons par calculer les coefficients de rang le plus élevé. Manuellement, cette méthode serait plus longue.

```
30 INPUT "BASE ";B
40 :
50 INPUT "Nombre ";N
60 :
70 FOR P=6 TO 0 STEP-1
80 : X=(B^P)
90 : Q=INT((N/X)+.001)
100 : N=N-X*Q
110 : PRINT Q;
120 NEXT P

RUN

Base? 2

Nombre? 5

000101
```

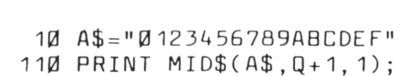
Hexadécimal :

Pour la conversion en **base 16,** il faut définir des symboles pour 10, 11, 12, ... 15.

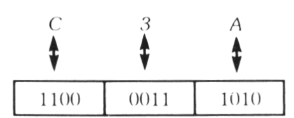
Choisissons A pour 10, B pour 11, ...

Pourquoi l'hexadécimal est-il utilisé en informatique?

Il permet de représenter de façon plus concise les nombres binaires; en effet, à chaque chiffre hexadécimal, il correspond 4 chiffres binaires :



```
10 A$="0123456789ABCDEF"
110 PRINT MID$(A$,Q+1,1);
```



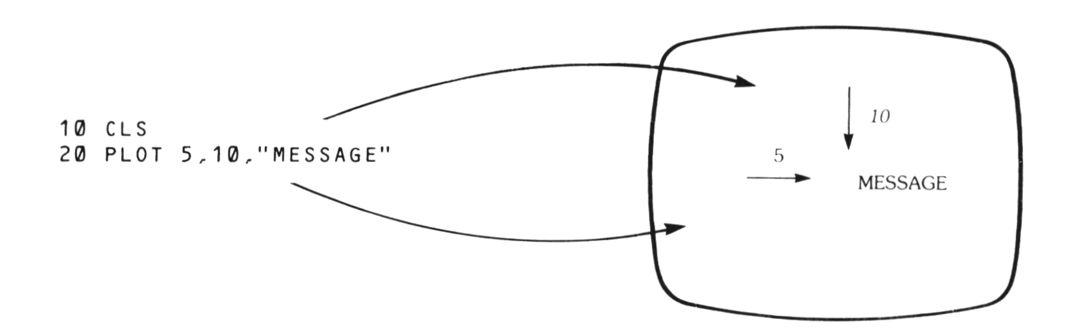
Nous pourrions également faire :

```
5 DIM S$(16)
10 DATA 0,1,2...A,B,C,D,E,F
20 FOR I=1 TO 16 : READ S$(I):NEXTI
110 PRINT S$(Q+1);
```

ADRESSAGE DIRECT ÉCRAN

PLOT X,Y,« Message »

L'adressage direct écran permet d'afficher un message à un endroit bien précis de l'écran (sans affecter le reste de l'écran).



Enseigne lumineuse :

Ce programme fait défiler un libellé sur l'écran. La vitesse de défilement dépend de la temporisation choisie.

```
5 REM    ENSEIGNE
6 :
20 CLS
30 L$="EDITIONS DU P.S.I........."
40 :
50 PLOT 5,10,L$
60 WAIT 20
70 L=LEN(L$):L$=RIGHT$(L$,1)+LEFT$(L$,L-1)
80 GOTO 50
```

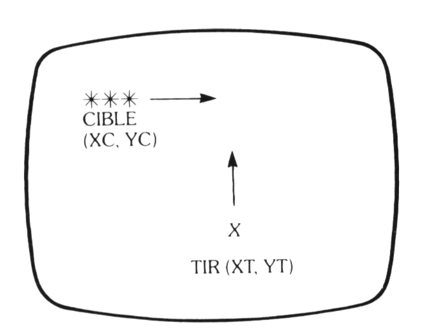
L$

E	DI . . . PS	I

PSI... ÉDITIONS

Tir sur cible

Une cible traverse l'écran de gauche à droite. La frappe d'une touche quelconque provoque un tir du bas de l'écran vers le haut.



En appuyant sur 'G' ou sur 'D', la trajectoire du tir est à −45 ou +45 degrés. Plusieurs tirs sont possibles.

```
10 REM       TIR SUR CIBLE
20 :
30 CLS:DP=0
40 VC=1:XC=2:YC=6:PLOT XC-1,YC,"***"
50 XT=24:YT=20:VT=-1:PLOT XT,YT,"X"
60 :
70 X$=KEY$:IF X$  ="" THEN 120     :REM Attente depart tir
80 DP=1:A=0                        :REM Angle tir nul
90 IF X$="G" THEN A=-.25
100 IF X$="D" THEN A=.25
110 IF YT<YC THEN XT=20:YT=15:VT=-1:PLOT XT,YT,"X"
120 REM ------------------ Avance cible ---------------
125 :
130 PLOT XC-1,YC," "              :REM Eff ancienne cible
140 XC=XC+VC
150 PLOT XC-1,YC,"***"
160 REM ------------------ Avance tir ------------------
170 IF DP=0 THEN 230
180 :
190 PLOT XT,YT," "
200 IF YT<YC THEN 230
210 YT=YT+VT:XT=XT+A:PLOT XT,YT,"X"
220 :
230 IF XC=>36   THEN PLOT 1,20,"PERDU":WAIT 300:GOTO 30
240 :
250 IF YT=YC AND XT=>XC-1 AND XT<=XC+1 THEN EXPLODE:END
260 :
270 WAIT 5
280 GOTO 70
```

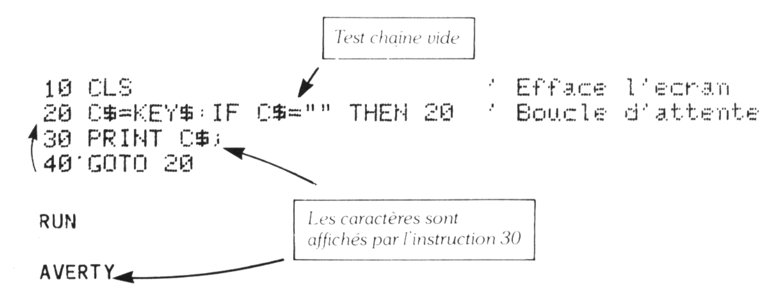
SAISIE D'UN CARACTÈRE AU CLAVIER

L'instruction INPUT ne permet pas au programme de vérifier la frappe des caractères au fur et à mesure de leur frappe mais seulement après que l'opérateur les ait validé par la touche <RETURN> ou <ENTER>.

L'instruction KEY$ le permet.

KEY$

Lit le CLAVIER EN PERMANENCE. Si aucun caractère n'a été frappé, la chaîne lue (C$ sur l'exemple) est vide. Le caractère frappé au clavier n'est affiché que si le programme l'a prévu (et non pas en 'local' par le clavier comme c'est le cas avec INPUT).




```
10 CLS                               ' Efface l'ecran
20 C$=KEY$:IF C$="" THEN 20          ' Boucle d'attente
30 PRINT C$;
40 GOTO 20

RUN

AVERTY
```

Le programme ci-dessous affiche le caractère dont le code est celui du caractère frappé plus 1.

```
10 CLS                               ' Efface l'ecran
20 C$=KEY$:IF C$="" THEN 20          ' Boucle d'attente
30 PRINT CHR$(ASC(C$)+1);
40 GOTO 20

RUN

BWFSUZ
```

Vous avez frappé AVERTY

Celui-ci n'affiche que les caractères compris entre « A » et « L ».

```
10 CLS                               ' Efface l'ecran
20 C$=KEY$:IF C$="" THEN 20          ' Boucle d'attente
30 IF C$<"A" OR C$>"L" THEN 20
40 PRINT C$;
50 GOTO 20

RUN

AE
```

Vous avez frappé AVERTY

Boucle d'attente

Attend que l'opérateur appuie sur une touche quelconque.

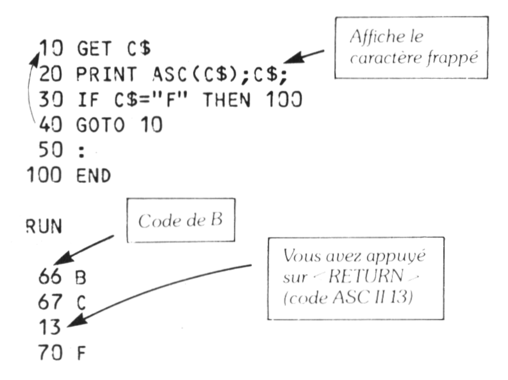
```
10 X$=KEY$:IF X$="" THEN 10        ' Boucle d'attente
20 PRINT "C'EST PARTI"
```

```
10 PRINT "Repondez O/N (vite) "
20 N=0
30 :
40 R$=KEY$:IF R$="O" THEN 100
50 N=N+1:IF N>1000 THEN PRINT "Trop tard":STOP
60 GOTO 40
70 :
100 REM suite
```

GET caractère

Lit un caractère au clavier dès sa frappe.

Celui-ci doit être affiché par le programme.




```
10 GET C$
20 PRINT ASC(C$);C$;
30 IF C$="F" THEN 100
40 GOTO 10
50 :
100 END

RUN

66 B
67 C
13
70 F
```

En voici un second exemple : **acquisition de nombres.**

Ce programme n'accepte que la frappe de chiffres. Si, par exemple, l'opérateur frappe une lettre, celle-ci n'est pas affichée et la sonnerie est activée.
Les caractères validés sont concaténés dans une chaîne L$.
Dès que l'opérateur appuie sur <RETURN> (code 13), le programme s'arrête.

ATTENTION : le programme présenté ne 'gère' pas le caractère d'annulation '←'(Code 127)
Celui-ci est concaténé à la chaîne L$ comme les autres caractères.

```
10 L$=""
15 :
20 GET C$
30 IF X$<"0" OR C$>"9" THEN PRINT CHR$(7);:GOTO 20
40 IF ASC(C$)=13 THEN 100      :REM retour chariot?
50 PRINT C$;
60 L$=L$+C$
70 GOTO 20
80 :
100 PRINT L$
110 PRINT VAL(L$)
```



```
RUN

314

314

314
```


Saisie d'une ligne avec GET.

```
10 REM  SAISIE D'UNE LIGNE AVEC GET
20 :
25 CLS
30 LIG$=""
40 :
50 GET C$
60 C=ASC(C$):L=LEN(LIG$)
70 IF C=13 THEN PRINT:GOTO 200
80 IF C<>127 THEN 120           :REM 127=DEL
90 :
100 IF L>0 THEN LIG$=LEFT$(LIG$,L-1):PRINT CHR$(127);:GOTO 50 ELSE
50
110 :
120 IF C<32 THEN PING:GOTO 50
130 LIG$=LIG$+C$
140 PRINT C$;
150 GOTO 50
160 :
200 PRINT LIG$
```

Test clavier avec PEEK (# 208) :

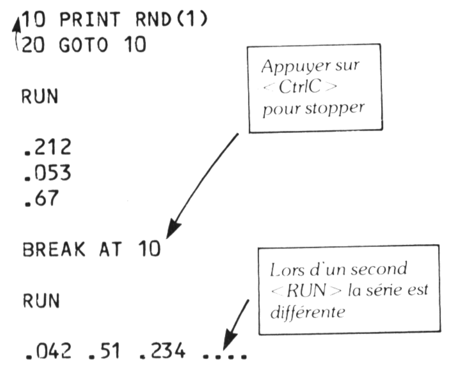
```
10 REM   Test clavier avec PEEK(#208) - Plus rapide -
20 REM
30 LORES0:X=10:Y=10:PLOT X,Y,17
35 :
40 C=PEEK(#208):IF C=56 THEN 40    :REM boucle d'attente
60 IF C=172 THEN IF X>1 THEN X=X-1   :REM gauche
70 IF C=188 THEN IF X<37 THEN X=X+1  :REM droite
80 IF C=156 THEN IF Y>1 THEN Y=Y-1
90 IF C=180 THEN IF Y<23 THEN Y=Y+1
100 PLOT X,Y,17
110 GOTO 40
```

LES NOMBRES ALÉATOIRES :

Les nombres aléatoires sont essentiellement utilisés pour les programmes de jeu ou d'éducation.

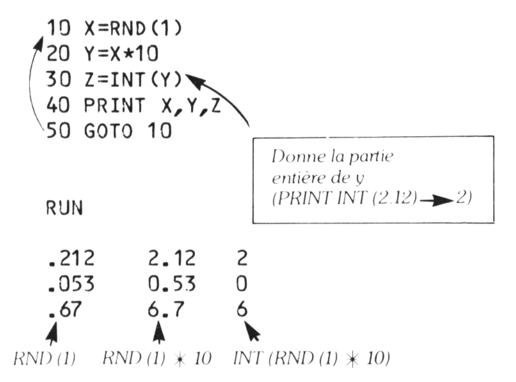
RND(X) pour X > 0 :

Fournit un nombre aléatoire compris entre 0 et 1 (1 exclus)



Pour obtenir des nombres entiers entre 0 et 9 par exemple, il faut :

- multiplier le nombre obtenu par 10
- prendre la partie entière du résultat, avec la fonction INT (X)



Pour obtenir un nombre entre 1 et 10, il suffit d'ajouter 1.

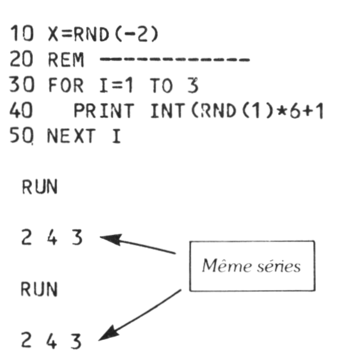
```
10 FOR N=1 TO 3
20  PRINT INT(RND(1)*10)+1
30 NEXT N

RUN

3  1 10
```

RND (X) pour X < 0 :

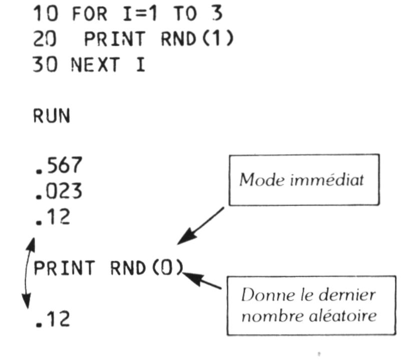
X négatif initialise une série aléatoire qui dépend de la valeur de X. Cette série est toujours la même pour une valeur de X.



Sans 10 X = RND (– 2), la série aléatoire après le second 'RUN' aurait été différente de celle obtenue après le premier 'RUN'.

RND (0)

Fournit le dernier nombre aléatoire.



Divisions :

Ce programme choisit 2 nombres au hasard, les affiche à l'écran, attend en réponse leur quotient puis vérifie que le résultat est juste.
Le niveau de difficulté est choisi au départ.

```
10 REM DIV          DIVISIONS
20 :
30 INPUT "Dividende maxi ? (ex 1000) ";DV
40 :
50 X=INT(RND(1)*DV)+1
60 Y=INT(RND(1)*DV/10)+1
70 :
80 PRINT:PRINT "Quel est le quotient de ";X;"par";Y;
90 INPUT "  Votre reponse? ";REP
100 IF INT(X/Y)=REP THEN PRINT:PRINT "C'est bon":GOTO 50
110 :
120 PRINT:PRINT "Le quotient de ";X;"par";Y "est";INT(X/Y)
130 GOTO 50
```

```
RUN

Dividende maxi ? (ex 1000)  100

Quel est le quotient de 45 par 6   Votre reponse? 7

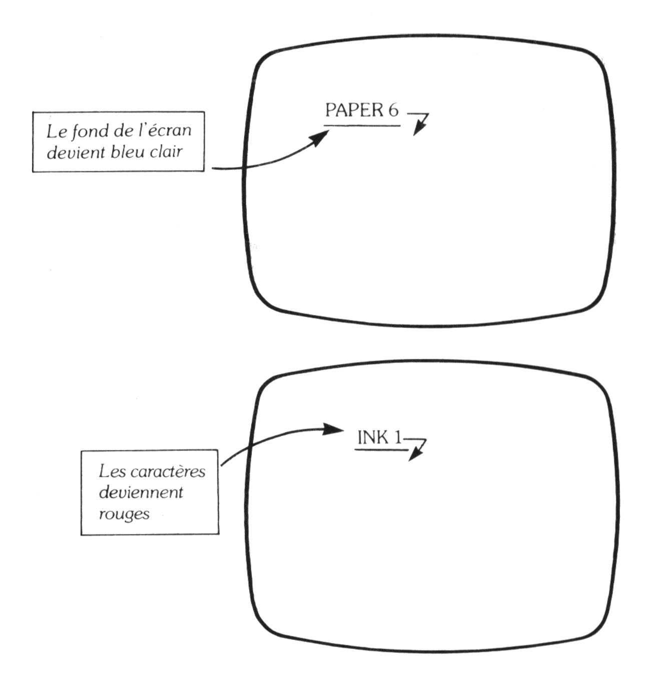
C'est bon


le reste se calculerait par: R=X-INT(X/Y)
```

CHOIX DES COULEURS

Les couleurs du fond de l'écran **(BACKGROUND)** et la couleur des caractères **(FOREGROUND)** se choisissent à l'aide de **PAPER** et **INK.** Les numéros de couleur sont compris entre 0 et 7. Au moment de la mise sous tension, le fond est blanc et la couleur des caractères noire.
En frappant PAPER 6, le fond devient bleu clair.

0 noir
1 rouge
2 vert
3 jaune
4 bleu
5 violet
6 bleu clair
7 blanc



INK et **PAPER** s'appliquent à l'écran entier. Nous verrons plus loin comment choisir une couleur de fond pour une partie d'écran seulement.

GRAPHIQUE BASSE RÉSOLUTION

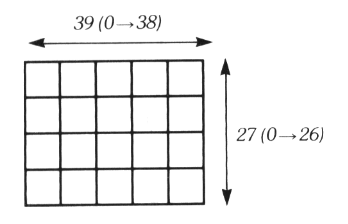
LORES 0
LORES 1
PLOT X,Y,chaîne

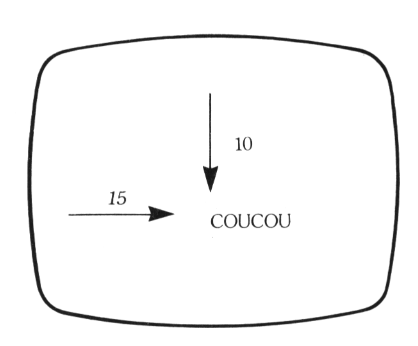
En graphique basse résolution, l'écran est divisé en 39 × 27 points

Les 2 modes **LORES 0** et **LORES 1** permettent d'afficher respectivement le jeu de caractères standards et le jeu de caractères dit alternés (ou semi graphiques).
Par exemple, A en alterné devient : ■ ■ .

PLOT X,Y,chaîne :

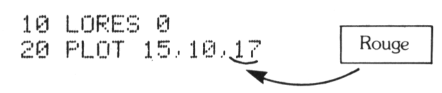
```
10 LORES 0
20 PLOT 15,10,"COUCOU"
```

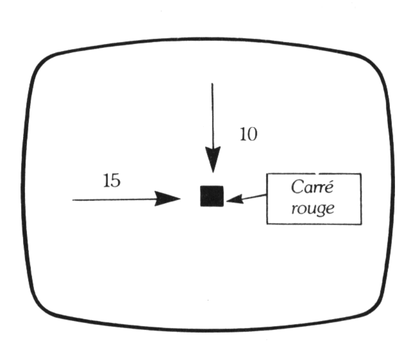




PLOT X,Y,code caractère :

Les codes 17 à 23 permettent de dessiner des fonds de couleur.



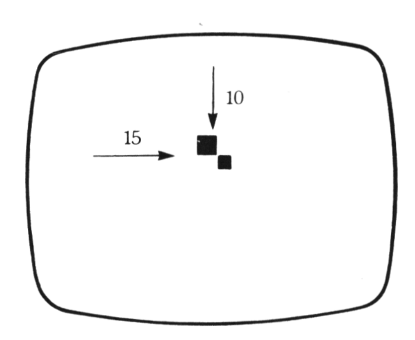


Différentes couleurs

16 noir
17 rouge
18 vert
19 jaune
20 bleu
21 violet
22 bleu clair
23 blanc

En choisissant LORES 1, on obtient à la place du caractère « A » le caractère ■ ■

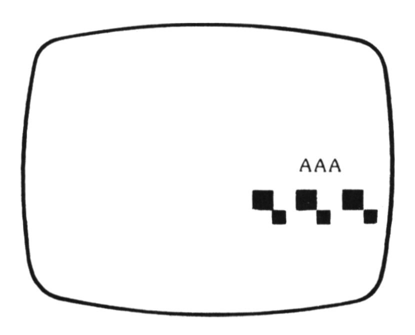
```
10 LORES 1
20 PLOT 15,10,"A"
```



Accès au jeu alterné en LORES 0

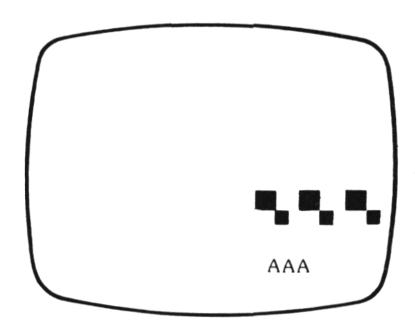
En **LORES 0,** on accède au jeu alterné à l'aide de CHR$(8)

```
10 LORES 0
20 X$="AAA"
30 M$=CHR$(9)+"AAA"+CHR$(8)
40 PLOT 10,15,X$
50 PLOT 10,16,M$
```



Sur **LORES 1,** l'accès au jeu de caractères standard se fait ainsi :

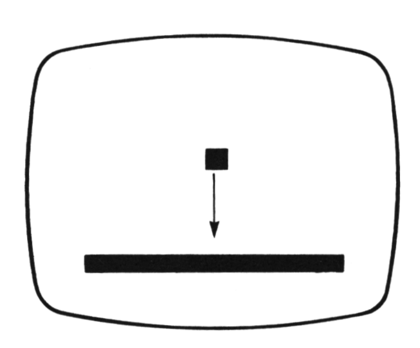
```
10 LORES 1
20 X$="AAA"
30 M$=CHR$(8)+X$+CHR$(9)
40 PLOT 10,15,X$
50 PLOT 10,16,M$
```



Pavé

Nous simulons une chute de pavé partant du haut de l'écran, le pavé est successivement dessiné puis effacé.

```
5 REM ------ PAVE ------
10 LORES 0
15 :                          REM Dessin bas
20 FOR X=1 TO 10 :PLOT X,24,17:NEXT X
25 :
30 FOR I=1 TO 24
40 :    PLOT 5,I-1,16       :REM Efface
50 :    PLOT 5,I,17         :REM Pave rouge
60 :    WAIT 5
70 NEXT I
80 EXPLODE
90 GOTO 90
```

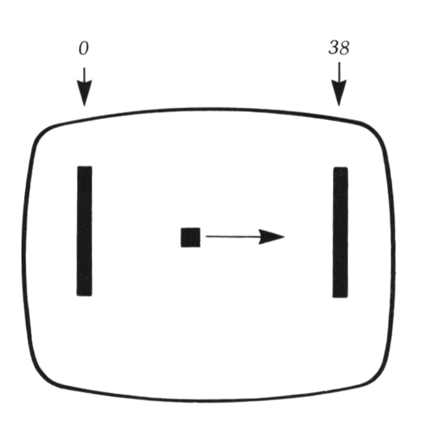


Ciel étoilé multicolore

```
10 REM   Ciel etoile multicolore
20 :
25 LORES0
30 FOR E=1 TO 100
40 :X=RND(1)*38:Y=RND(1)*23
50 :C=RND(1)*7+1
60 :PLOT X,Y,16+C
70 NEXT E
```

La balle qui rebondit :

Ce programme fait rebondir une balle entre 2 murs.




```
5 REM --- La balle qui rebondit ---
6 :
10 LORES 0
20 FOR Y=1 TO 20
30 : PLOT 0,Y,20:PLOT 38,Y,20
40 NEXT Y
50 :
60 X=10:Y=10
70 VITESSE=-1
80 X=X+VITESSE
90 IF X<1 OR X>37 THEN 150
100 :
110 PLOT X,Y,17
120 PLOT X,Y,16
130 GOTO 80
140 :
150 VITESSE=-VITESSE
160 PING
170 GOTO 80
```

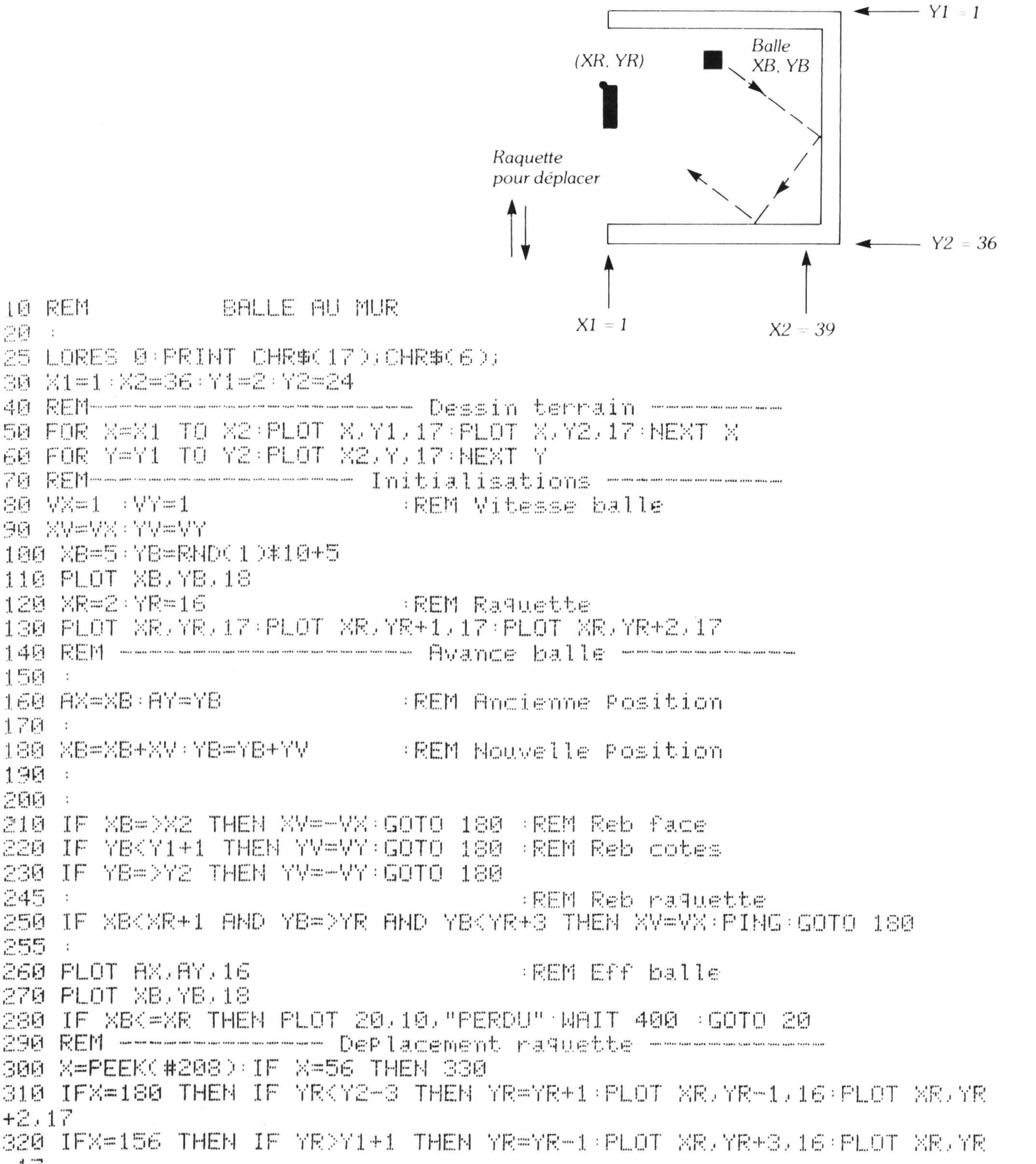
Celui-ci fait rebondir la balle entre les 4 murs.

```
10 REM BILLARD
20 :
30 LORES 0
32 XB=2:XH=38
34 YB=2:YH=23
40 :
50 FOR X=XB TO XH
60 : PLOT X,YB,20
70 : PLOT X,YH,20
80 NEXT X
90 :
100 FOR Y=YB TO YH
110 : PLOT XB,Y,20:PLOT XH,Y,20
120 NEXT Y
130 :
140 XP=XB+INT(RND(1)*6)+1:YP=YB+INT(RND(1)*6)+1
150 VX=1:VY=1
160 :
170 IF XP<XB+1 OR XP>XH-1 THEN VX=-VX:PING:GOTO 220
180 IF YP<YB+1 OR YP>YH-1 THEN VY=-VY:PING:GOTO 220
190 :
200 PLOT XP,YP,19
210 WAIT 5
215 PLOT XP,YP,16
218 :
220 XP=XP+VX:YP=YP+VY
230 GOTO 170
```

Squash

Une balle rebondit sur 3 murs.
Le joueur doit la faire rebondir sur la raquette qu'il déplace avec les flèches ↑ ↓

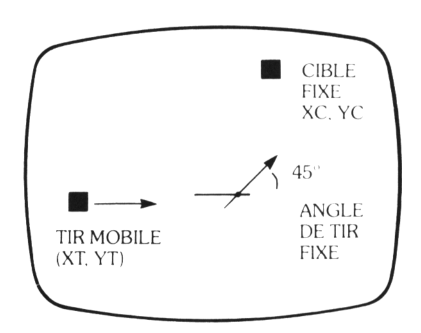
Remarque : Nous aurions pu inverser VX et VY à chaque rebond en faisant VX = −VX et VY = −VY. Mais pour des raisons d'arrondi, il pourrait se produire des faux rebonds.




```
10 REM            BALLE AU MUR
20 :
25 LORES 0:PRINT CHR$(17);CHR$(6);
30 X1=1:X2=36:Y1=2:Y2=24
40 REM------------------------- Dessin terrain ----------
50 FOR X=X1 TO X2:PLOT X,Y1,17:PLOT X,Y2,17:NEXT X
60 FOR Y=Y1 TO Y2:PLOT X2,Y,17:NEXT Y
70 REM---------------------- Initialisations -------------
80 VX=1 :VY=1                :REM Vitesse balle
90 XV=VX:YV=VY
100 XB=5:YB=RND(1)*10+5
110 PLOT XB,YB,18
120 XR=2:YR=16               :REM Raquette
130 PLOT XR,YR,17:PLOT XR,YR+1,17:PLOT XR,YR+2,17
140 REM ----------------------- Avance balle -------------
150 :
160 AX=XB:AY=YB              :REM Ancienne Position
170 :
180 XB=XB+XV:YB=YB+YV        :REM Nouvelle Position
190 :
200 :
210 IF XB=>X2 THEN XV=-VX:GOTO 180 :REM Reb face
220 IF YB<Y1+1 THEN YV=VY:GOTO 180 :REM Reb cotes
230 IF YB=>Y2 THEN YV=-VY:GOTO 180
245 :                              :REM Reb raquette
250 IF XB<XR+1 AND YB=>YR AND YB<YR+3 THEN XV=VX:PING:GOTO 180
255 :
260 PLOT AX,AY,16                  :REM Eff balle
270 PLOT XB,YB,18
280 IF XB<=XR THEN PLOT 20,10,"PERDU":WAIT 400 :GOTO 20
290 REM -------------- Deplacement raquette -------------
300 X=PEEK(#208):IF X=56 THEN 330
310 IFX=180 THEN IF YR<Y2-3 THEN YR=YR+1:PLOT XR,YR-1,16:PLOT XR,YR
+2,17
320 IFX=156 THEN IF YR>Y1+1 THEN YR=YR-1:PLOT XR,YR+3,16:PLOT XR,YR
,17
330 GOTO 160
```

Tir sur cible

Voici un programme de tir. Cette fois la cible est fixe et le tir mobile (CHAR par exemple). L'angle de tir est de 45 degrés ; il pourrait être changé en modifiant la vitesse VT.

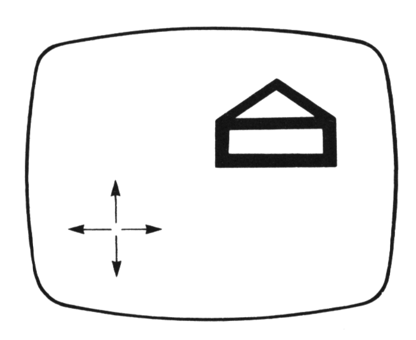



```
10 REM ------ TIR SUR CIBLE ------
20 :
30 LORES 0                          :REM mode graphique
35 DP=0
40 XC=INT(RND(1)*5)+30:YC=5
50 PLOT XC,YC,17
60 VT=-1                            :REM vitesse tir
70 XT=1:YT=20                       :REM coordonnees tir
80 :
90 PLOT AX,AY,16                    :REM efface ancienne position
100 PLOT XT,YT,17
110 AX=XT:AY=YT                     :REM ancienne position
120 IF YT<YC THEN PLOT 10,10,"RATE":WAIT 400:GOTO 20
125 IF XT>37 THEN PLOT 10,10,"RATE":WAIT 400:GOTO 20
130 IF XT=XC AND YC=YT THEN EXPLODE:PLOT 10,10,"GAGNE":END
140 :
150 X$=KEY$:IF X$<>"" THEN DP=1:ZAP   :REM Depart tir
160 IF DP=0 THEN 200
170 :
180 YT=YT+VT
190 :
200 XT=XT+1:GOTO 90
```

Télécran

Principe : En appuyant sur les flèches ↑ ↓ ← →, vous déplacez un « point » qui laisse une « trace » sur son passage ce qui vous permet de dessiner « manuellement ».
Si vous avez appuyé sur « L », le déplacement s'effectue sans laisser de trace, permettant ainsi de dessiner une figure en plusieurs parties.

Améliorations possibles : Enregistrement du dessin dans une table pour duplication avec échelle.



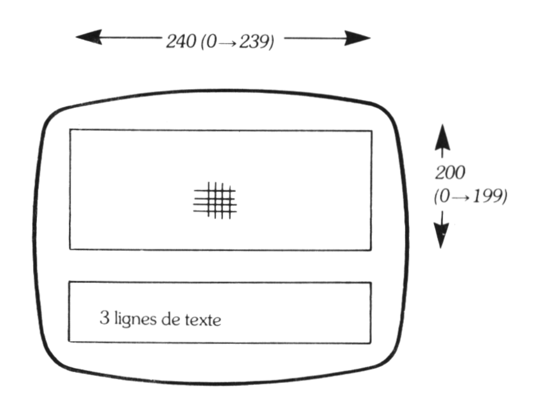
```
10 REM     TELECRAN
20 :
25 LORES 0
30 X=10:Y=10            :REM Coordonnees depart
40 :
50 PLOT 30,18,"B:BAISSER"
60 PLOT 30,19,"L:LEVER"
70 :
80 C$=KEY$:IF C$<>"" THEN 130  :REM Attente frappe clavier
90 PLOT X,Y,17
100 PLOT X,Y,16                :REM Curseur clignotant
110 GOTO 80
120 :
130 IF L=0 THEN PLOT X,Y,17    :REM Allume en rouge
140 :
150 C=ASC(C$)
160 IF C=8 THEN IF X>0 THEN X=X-1  :REM Gauche
170 IF C=9 THEN IF X<38 THEN X=X+1 :REM Droite
180 IF C=10 THEN IF Y<24 THEN Y=Y+1
190 IF C=11 THEN IF Y>0 THEN Y=Y-1
200 :
210 IF C$="L" THEN L=1      :REM Lever
220 IF C$="B" THEN L=0      :REM Baisser
230 GOTO 80
```

GRAPHIQUES HAUTE RÉSOLUTION

HIRES — TEXT
CURSET X, Y, couleur
CURMOV DX, DY, couleur
DRAW DX, DY, couleur
CIRCLE rayon, couleur
POINT X, Y
PATTERN X
FILL hauteur, longueur, couleur
CHAR code, jeu caractère, couleur

L'instruction HIRES permet d'accéder au mode 'haute résolution'
Le retour en mode normal (TEXT) se fait par **l'instruction TEXT.**
En haute résolution, l'écran est divisé en 240*200 points.

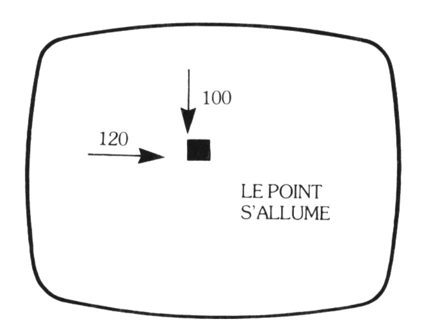
HIRES initialise la couleur background (de fond) en noir et la couleur foreground en blanc.
3 lignes sont réservées pour le texte en bas de l'écran.



CURSET X,Y,type couleur

Le positionnement du 'curseur' se fait par CURSET. C'est un positionnement en absolu.
'type couleur' peut être égal à 0, 1, 2, 3.
Pour 'type couleur' égal à 1, CURSET allume le point spécifié.

```
10 HIRES
20 CURSET 120,100,1
```



Pour revenir en mode texte, nous frappons en mode direct :

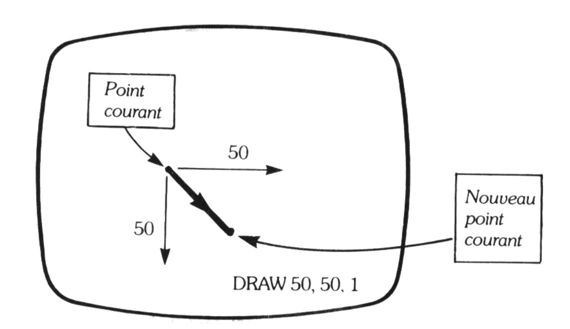
TEXT

Nous développerons plus loin la notion de 'type de couleur'.

DRAW DX,DY,type couleur :

Trace une droite à partir du point courant (position du curseur). Les paramètres DX et DY spécifient des déplacements relatifs au point courant.

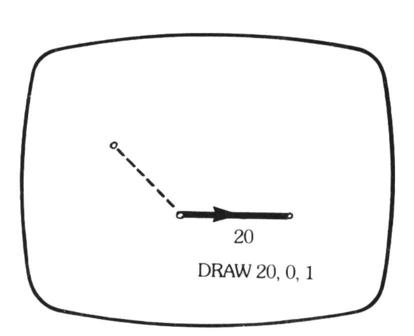
```
10 HIRES
20 CURSET 120,100,1
30 DRAW 50,50,1
```



Le point courant devient (120 + 50), (100 + 50).
En ajoutant au programme :

```
40 DRAW 20,0,1
```

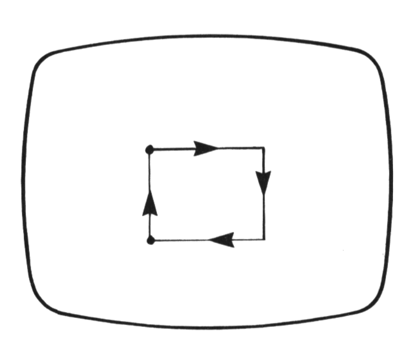
On obtient le tracé suivant.



TEXT permet de revenir au mode normal.

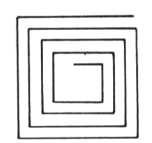
Par exemple, dessinons un carré :

```
10 HIRES
20 CURSET 100,100,1
30 :
40 DRAW 50,0,1
50 DRAW 0,50,1
60 DRAW -50,0,1
70 DRAW 0,-50,1
```



Et maintenant... un escargot :

```
10 HIRES:INK3
15 CURSET 100,100,0
20 :
30 FOR N=1 TO 8*16 STEP 8
40 : DRAW N      ,0      ,1
50 : DRAW 0      ,N+2    ,1
60 : DRAW -(N+4),0      ,1
70 : DRAW 0      ,-(N+6),1
80 NEXT N
```



```
20 REM Spirale
30 REM
40 HIRES:INK3
50 D=10:CURSET 100,100,1
60 FOR ANG=0 TO 22*PI STEP PI/2
70 :CX=COS(ANG):IF CX>-.01 AND CX<.01 THEN CX=0
80 :CY=SIN(ANG):IF CY>-.01 AND CY<.01 THEN CY=0
90 :DX=D*CX:DY=D*CY:DRAW DX,DY,1
100 :D=D+2
110 NEXT ANG
```

Types de couleur :

4 types de couleur peuvent être spécifiés dans les instructions graphiques :

— Foreground (1) : DRAW 50,50,1

Dessine avec la couleur foreground (choisie par INK couleur)

— Background (0) : DRAW 50,50,0

Dessine avec la couleur de fond (choisie par PAPER couleur)
Peut servir à effacer un tracé fait en couleur foreground.

— Inverse (2) : DRAW 50,50,2

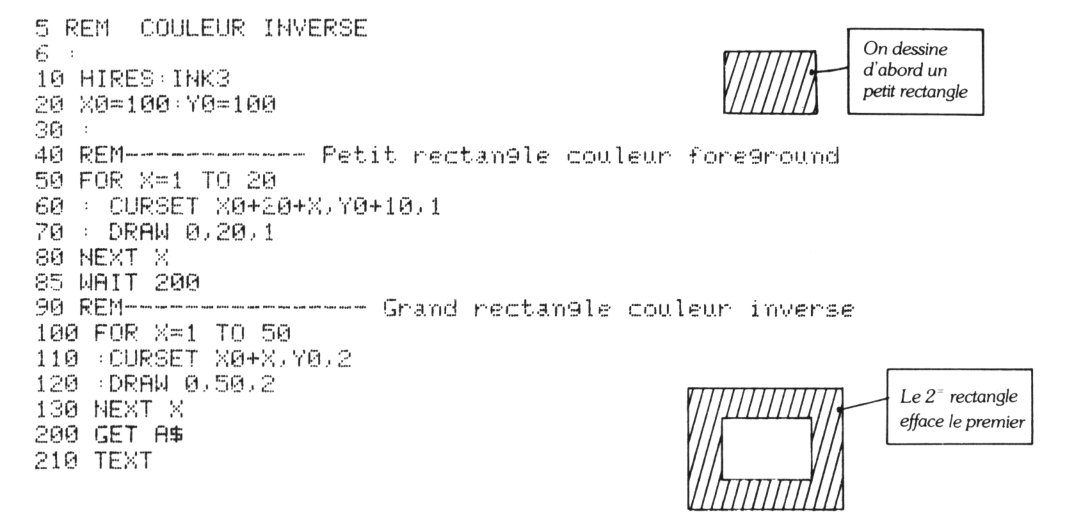
Dessine en couleur inverse :
- Si le tracé n'existait pas, il est fait en couleur foreground.
- Si le tracé a lieu sur une zone déjà coloriée, il se fait en couleur de fond.

— Nul (3) : DRAW 50,50,3

Ne dessine pas.
Ci-dessous, nous dessinons un anneau de la façon suivante :

- Nous dessinons d'abord un petit rectangle en couleur foreground.
- Nous superposons un grand rectangle en couleur inverse.
  Le rectangle central est 'effacé'. Ainsi, il reste un anneau.

(Nous aurions pu également dessiner le rectangle central en couleur de fond après avoir dessiné le grand rectangle).



```
5 REM  COULEUR INVERSE
6 :
10 HIRES:INK3
20 X0=100:Y0=100
30 :
40 REM------------ Petit rectangle couleur foreground
50 FOR X=1 TO 20
60 : CURSET X0+20+X,Y0+10,1
70 : DRAW 0,20,1
80 NEXT X
85 WAIT 200
90 REM---------------- Grand rectangle couleur inverse
100 FOR X=1 TO 50
110 :CURSET X0+X,Y0,2
120 :DRAW 0,50,2
130 NEXT X
200 GET A$
210 TEXT
```

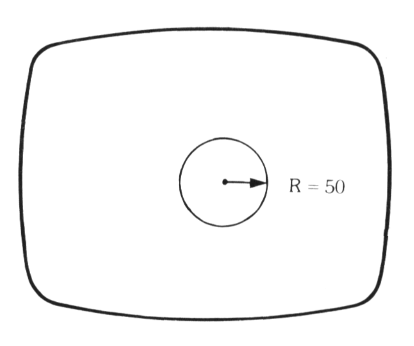
CURMOV DX,DY,type couleur :

Agit comme CURSET X,Y,type couleur, mais les paramètres spécifiés sont des déplacements relatifs au point courant.

CIRCLE rayon,type couleur :

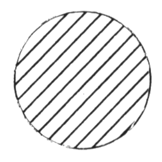
Trace un cercle dont le centre est le point courant.

```
10 HIRES
20 CURSET 100,100,1
30 CIRCLE 50,1
```



Pour tracer un cercle plein, on fait :

```
10 HIRES
20 CURSET 100,100,1
30 FOR R=1 TO 30
40 : CIRCLE R,1
50 NEXT R
```



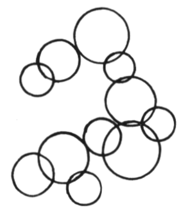
On pourra ajouter :

```
100 FOR B=0 TO 7
110 :PAPER B:WAIT 100
120 :FOR F=0 TO 7
130 :INK F:WAIT 100
140 :NEXT F
150 NEXT B
```

Cercles aléatoires :

Des cercles de rayon aléatoire sont affichés aléatoirement sur l'écran.

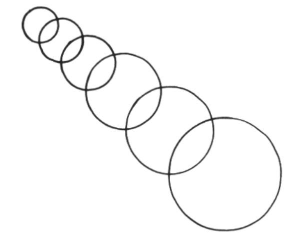
```
10 HIRES
15 INK3
20 :
30 FOR N=1 TO 20
40 : X=RND(1)*200+20
50 : Y=RND(1)*150+20
55 : R=10+RND(1)*10
60 : CURSET X,Y,0
70 : CIRCLE R ,1
80 NEXT N
```



Cône :

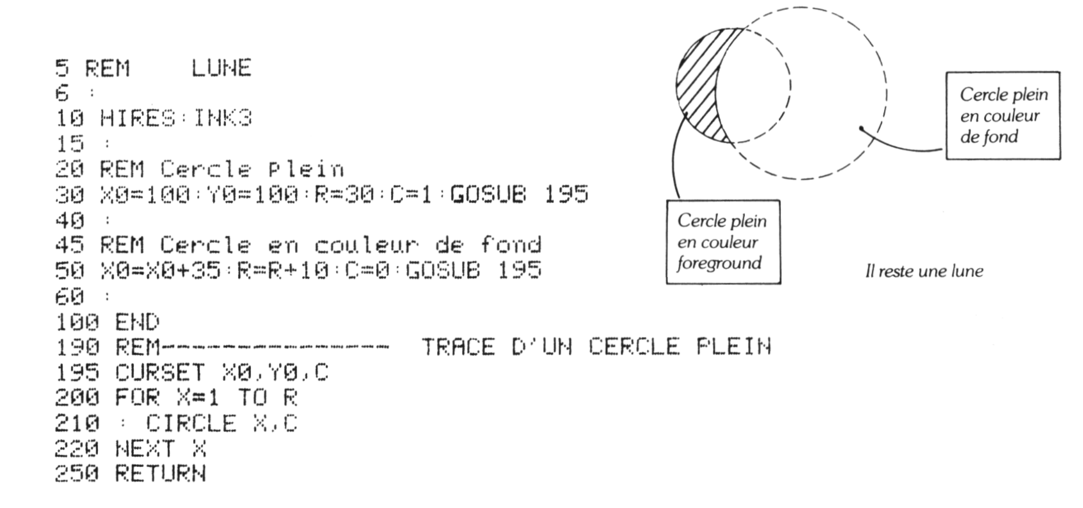
Dessine un cône en perspective.

```
10 HIRES
15 INK3
20 :
30 FOR R=5 TO 50 STEP 4
60 : CURSET 30+R,30+R*2,1
70 : CIRCLE R,1
80 NEXT R
```



Croissant de lune

Un dessin de lune est obtenu par superposition d'un cercle en couleur BACKGROUND sur un cercle en couleur FOREGROUND.




```
5 REM     LUNE
6 :
10 HIRES:INK3
15 :
20 REM Cercle Plein
30 X0=100:Y0=100:R=30:C=1:GOSUB 195
40 :
45 REM Cercle en couleur de fond
50 X0=X0+35:R=R+10:C=0:GOSUB 195
60 :
100 END
190 REM----------------  TRACE D'UN CERCLE PLEIN
195 CURSET X0,Y0,C
200 FOR X=1 TO R
210 : CIRCLE X,C
220 NEXT X
250 RETURN
```

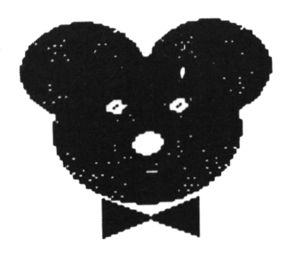
Chat



```
5 REM     CHAT
6 :
10 HIRES
12 INK3
15 REM ----------- Cercles:centre/rayon
20 DATA 100,80,10   :REM Yeux
30 DATA 140,80,10
40 DATA 120,100,50 :REM Tete
50 DATA 120,105,3   :REM Nez
55 :
60 DATA 40,7        :REM Moustaches
65 DATA 40,-7
70 DATA 40,0
75 :
77 DATA 120,125     :REM Bouche
80 DATA 10,-8
81 :
82 DATA 80,40       :REM Oreilles
83 DATA 12,17
84 DATA -5,40
85 :
86 DATA 160,40
87 DATA -12,17
88 DATA 5,40
89 REM ---------- Tete,yeux,nez ------
100 FOR P=1 TO 4
110 :    READ X,Y,R
120 :    CURSET X,Y,1
130 :    CIRCLE R,1
140 NEXT P
150 REM -------- Moustaches ----------
200 FOR M=1 TO 3
210 :    READ DX,DY
220 :    CURSET X,Y,1
230 :    DRAW DX,DY,1
240 :    CURSET X,Y,1
250 :    DRAW -DX,-DY,1
260 NEXT M
270 REM ------- Bouche ---------------
300 READ X,Y
310 READ DX,DY
320 CURSET X,Y,1
330 DRAW DX,DY,1
340 CURSET X,Y,1
350 DRAW -DX,DY,1
```

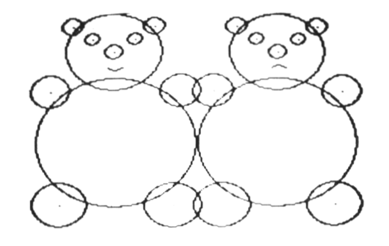
```
360 REM ------- Oreilles ------------
400 READ X,Y
410 CURSET X,Y,1
420 READ DX,DY
430 DRAW DX,DY,1
440 CURSET X,Y,1
450 READ DX,DY
460 DRAW DX,DY,1
470 :
490 READ X,Y:CURSET X,Y,1
500 READ DX,DY:DRAW  DX,DY,1
510 CURSET X,Y,1
520 READ DX,DY:DRAW DX,DY,1
1000 GET A$
1020 TEXT
```

Ours chic



```
5 REM       OURS  CHIC
6 :
10 HIRES:INK6
20 REM----------------------- Tete --------
25 X0=100:Y0=100
30 CURSET X0 ,Y00,1
40 FOR R=1 TO 30:CIRCLE R,1:NEXT R
70 REM ---------------------- Oreilles ----
80 CURSET X0-20,Y0-25,1
90 FOR R=1 TO 20:CIRCLE R,1:NEXT R
100 CURSET X0+20,Y0-25,1
110 FOR R=1 TO 20:CIRCLE R,1:NEXT R
120 REM---------------------- Truffe ------
130 CURSET X0,Y0+5,2
140 FOR R=1 TO 5:CIRCLE R,0:NEXT R
150 REM------------------ Yeux ------------
170 CURSET X0-10,Y0-10,1
180 FOR R=2 TO 3:CIRCLE R,0:NEXT R
190 :
200 CURSET X0+10,Y0-10,1
210 FOR R=2 TO 3:CIRCLE R,0:NEXT R
220 REM------------------ Papillon --------
230 FOR Y=-8 TO 8
240 : CURSET X0,Y0+35,1:DRAW -15,Y,1
260 : CURSET X0,Y0+35,1:DRAW +15,Y,1
280 NEXT Y
290 :
300 CURSET X0-1,Y0+16,0:DRAW 3,0,0
```

Ours jumeaux



```
10 HIRES:INK3
15 :                   :REM CENTRE/RAYON
20 DATA 120,100,50     :REM CORPS
30 DATA 120,30,30      :REM TETE
40 DATA 85,148,18      :REM PATTES
50 DATA 155,148,18
60 DATA 80,60,13
70 DATA 160,60,13
80 DATA 95,10,7        :REM OREILLES
90 DATA 145,10,7
100 DATA 107,20,5      :REM YEUX
110 DATA 133,20,5
120 DATA 120,30,6      :REM NEZ
130 REM------------------------- 1ER OURS
140 FOR P=1 TO 11
150 :    READ X1,Y1,R
160 :    CURSET X1-50,Y1,1
170 :    CIRCLE R,1
175 NEXT P
177 CURSET 65,42,3
178 DRAW 5,+2,1:DRAW 5,-2,1
180 REM----------------------- 2EME OURS
181 RESTORE
182 FOR P=1 TO 11
183 :   READ X1,Y1,R
184 :   CURSET X1+50,Y1,1
185 :   CIRCLE R,1
187 NEXT P
188 CURSET 165,42,3
189 DRAW 5,-2,1:DRAW 5,2,1
190 :
```

Dessin d'un cube :

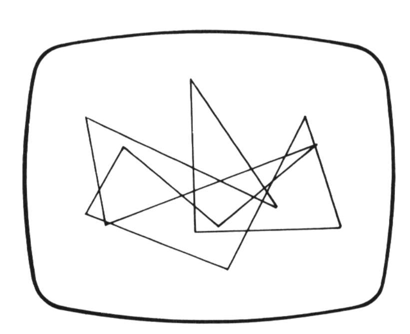
Les composantes de 3 vecteurs sont stockées dans 2 tables DX () et DY (). Nous calculons les coordonnées des différents sommets à partir desquels nous traçons les arêtes.

```
5 REM       DESSIN D'UN CUBE
6 :
10 X0=100:Y0=100
20 NC=3
30 REM -------- Composantes ---------------
40 DX(0)=50:DY(0)=0
50 DX(1)=0:DY(1)=50
60 DX(2)=-20:DY(2)=-20
70 :
80 HIRES
85 REM---------- Sommet en binaire ----------
90 FOR S=0 TO (2^NC)-1
100 :N=S
110 :FOR I=NC-1 TO 0 STEP-1
120 :   P(I)=INT(N/(2^I))
130 :   N=N-P(I)*(2^I)+.0001
140 :NEXT I
150 :REM------------CALCUL coordonnees sommet
160 :X(S)=X0:Y(S)=Y0
170 :FOR I=0 TO NC-1
180 :   IF P(I)=0 THEN 210
190 :   X(S)=X(S)+DX(I)
200 :   Y(S)=Y(S)+DY(I)
210 :NEXT I
220 :REM------------------ TRACE des aretes
230 :FOR I=0 TO NC-1
240 :   IF P(I)<>0 THEN 270
250 :   CURSET X(S),Y(S),1
260 :   DRAW DX(I),DY(I),1
270 :NEXT I
272 : PRINT "S=";S,P(2),P(1),P(0)
275 :WAIT 200
280 NEXT S
```

Dessin aléatoire :

Des points sont choisis au hasard. Nous les relions entre eux par des droites.

```
5 REM     DESSIN ALEATOIRE
6 :
10 HIRES
15 INK5
17 CURSET 10,10,1
20 FOR P=1 TO 50
30 :X=INT(RND(1)*150)+10
40 : Y=INT(RND(1)*150)
50 : DRAW X-XA,Y-YA,1
60 : XA=X:YA=Y
70 NEXT P
```



Dessin avec échelle :

Dessine une figure définie par des déplacements successifs.
En adaptant l'échelle, on peut reproduire le dessin avec des tailles différentes.

X0, Y0

Echelle 1

Echelle 5

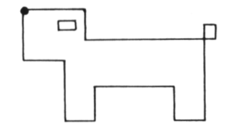
DÉPLACEMENT
DY DX

```
10 REM ----- DESSIN AVEC ECHELLE ---
20 :
30 DATA 0,10,  10,0,  0,20,  10,0
40 DATA 0,-10, 20,0,  0,10
50 DATA 10,0,  0,-20, 3,0,   0,-3
60 DATA -3,0,  0,+3 ,-30,0,  0,-10
70 DATA -20,0
80 DATA 99,99
90 :
100 HIRES : INK 3
110 X0=50:Y0=50:ECH=1
120 GOSUB 250          :REM 1er dessin ----
130 :
140 RESTORE
150 X0=120:Y0=70:ECH=.5
160 GOSUB 250          :REM 2eme dessin ---
170 GET X$
180 TEXT
190 END
200 REM ---- SOUS PROGRAMME DESSIN ----------------
220 :
250 FOR P=1 TO 1000
260 :READ DX:READ DY:IF DX=99 AND DY=99 THEN 320
270 :X1=DX*ECH
280 :Y1=DY*ECH
290 :CURSET X0,Y0,1:DRAW X1,Y1,1
300 :X0=X0+X1:Y0=Y0+Y1
310 NEXT P
320 RETURN
```

Les figures dessinées par ce programme ne doivent pas présenter de discontinuité.

Pour ajouter un œil au chien, nous pourrions indiquer qu'il y a discontinuité en codant 0,0 dans les DATAS :

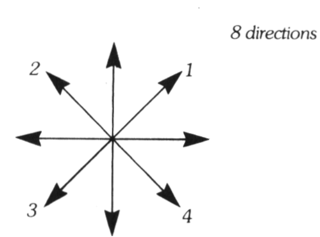
```
75 DATA 0,0,  10,3,  3,0
76 DATA 0,3,  -3,0,  0,-3
```



```
265 IF DX=0 AND DY=0 THEN READ DX:READ DY:X0=X0+DX*ECH:Y0=Y0+DY*ECH:GOTO
```

Super télécran :

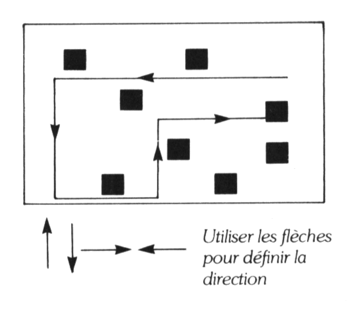
Nous reprenons le programme Télécran basse résolution en y ajoutant 4 autres directions. Les chiffres 1, 2, 3 et 4 servent à déplacer le curseur en diagonale.
Naturellement, on pensera à utiliser la touche « REPEAT ».




```
10 REM      TELECRAN (HAUTE RESOLUTION)
12 :
13 REM    8 Directions: 4 fleches+ 1 2 3 4
14 :
20 :
25 HIRES
30 X=100:Y=100        :REM Coordonnees depart
40 :
70 :
80 C$=KEY$:IF C$<>"" THEN 130  :REM Attente frappe clavier
90 CURSET X,Y,1
100 CURSET X,Y,0
110 GOTO 80
120 :
130 IF L=0 THEN CURSET X,Y,1   :REM Allume
140 :
150 C=ASC(C$)
160 IF C=8 THEN X=X-1  :REM Gauche
170 IF C=9 THEN  X=X+1 :REM Droite
180 IF C=10 THEN  Y=Y+1
190 IF C=11 THEN  Y=Y-1
192 IF C=49 THEN X=X+1:Y=Y-1
194 IF C=50 THEN X=X-1:Y=Y-1
196 IF C=51 THEN X=X+1:Y=Y+1
198 IF C=52 THEN X=X-1:Y=Y+1
200 :
210 IF C$="L" THEN L=1      :REM Lever
220 IF C$="B" THEN L=0      :REM Baisser
230 GOTO 80
```

Evitez l'obstacle!

Un point se déplace à travers des obstacles. Vous devez adapter la direction pour éviter les obstacles.


Utiliser les flèches pour définir la direction

```
10 REM       OBSTACLE
20 :
25 HIRES:INK3
26 INPUT "NIVEAU (1,2,..) ";N
27 IF N=0 THEN N=1
28 REM--------------------- Cadre
30 CURSET 15,15,1
40 DRAW 200,0,1
50 DRAW 0,150,1
60 DRAW -200,0,1
70 DRAW 0,-150,1
80 RE--------------- Obstacles
100 FOR P=1 TO N*10
110  : X=RND(1)*190+20
120  : Y=RND(1)*140+20
130  : CURSET X,Y,1
140  : CURSET X+1,Y,1
150  : CURSET X,Y+1,1
160  : CURSET X+1,Y+1,1
170 NEXT P
180 REM ---------- Coordonees depart
200 X=RND(1)*190+20
210 Y=RND(1)*140+20
220 IF POINT(X,Y)=-1 THEN 200
230 REM ---------- Attente depart
240 C$=KEY$
242 IF C$<>"" THEN GOSUB 260:GOTO 315
243 CURSET X,Y,1:WAIT 10
245 CURSET X,Y,0
248 GOTO 240
250 REM ------- Actualisation X,Y
260 C=ASC(C$)
270 IF C=10 THEN DX=0:DY=1
280 IF C=11 THEN DX=0:DY=-1
290 IF C=8  THEN DX=-1:DY=0
300 IF C=9  THEN DX=1:DY=0
305 RETURN
310 REM-------------------------------
315 P=0
317 :
320 C$=KEY$:IF C$<>"" THEN GOSUB 260
330 :
335 P=P+1
340 X=X+DX:Y=Y+DY
350 IF POINT(X,Y)=-1 THEN 400
355 CURSET X,Y,1
360 GOTO 320
370 REM-------------------------------
400 EXPLODE
410 PRINT P;"POINTS";
420 WAIT 300:GOTO 25
```

Tracé de cercles ou d'ellipses :

Un cercle peut se tracer point par point ou par segments de droites.
Naturellement, le tracé est beaucoup plus lent que s'il était effectué directement par une instruction BASIC, telle que **l'instruction CIRCLE.**

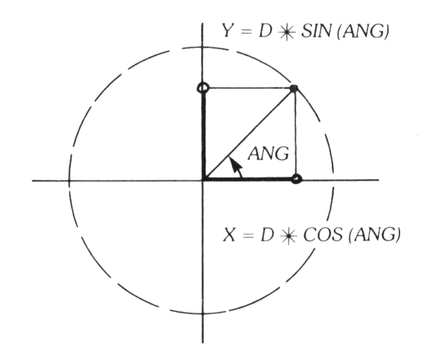
Nous calculons les coordonnées X et Y d'un cercle de rayon R par :

```
X=R*COS(ANG)
Y=R*SIN(ANG)
```

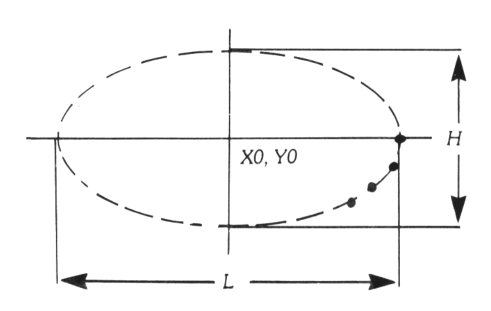
En faisant varier ANG de 0 à 6.28 radians.

Pour une ellipse, X et Y sont donnés par :

```
X=(L/2)*COS(ANG)
Y=(H/2)*SIN(ANG)
```




```
10 REM ELLIPSE
20 :
30 HIRES:INK3
40 X0=100:Y0=100
50 L=100:H=100
60 :
100 FOR ANG=0 TO 2*PI STEP.1
110 : X=X0+(L/2)*COS(ANG)
120 : Y=Y0+(H/2)*SIN(ANG)
130 : CURSET X,Y,1
140 NEXT ANG
150 :
160 GET X$
170 TEXT
```



En faisant H = L, nous obtenons bien sur un cercle de rayon L/2.

Tracé de figures inscrites dans un cercle

Nous présentons ici le tracé de figures inscrites dans un cercle ou une ellipse (triangle, carré, losange,...)

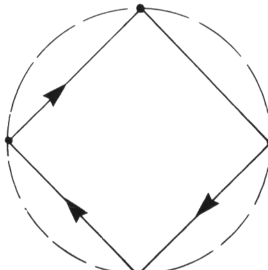
triangle :

```
10 HIRES
20 X0=100:Y0=100
30 H=130:L=100
40 NCOT=3               :REM Nombre de cotes
50 GOSUB 100
55 END
60 :
90 REM------------------- SPGM ---------------
100 XA=X0+L/2:YA=Y0
110 :
120 FOR C=1 TO NCOT
125 :  ANG=C*2*PI/NCOT
130 :  X=X0+L/2*COS(ANG)
140 :  Y=Y0+H/2*SIN(ANG)
145 :
150 :  CURSET XA,YA,1
155 :  DRAW X-XA,Y-YA,1
160 :  XA=X:YA=Y
170 NEXT C
180 RETURN
```

X0 Y0

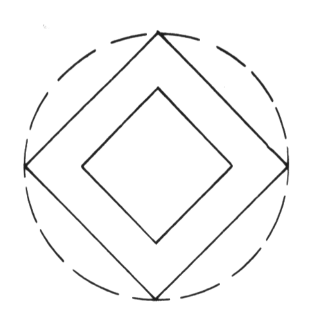
carré :

Il suffit de faire : 40 NCOT = 4



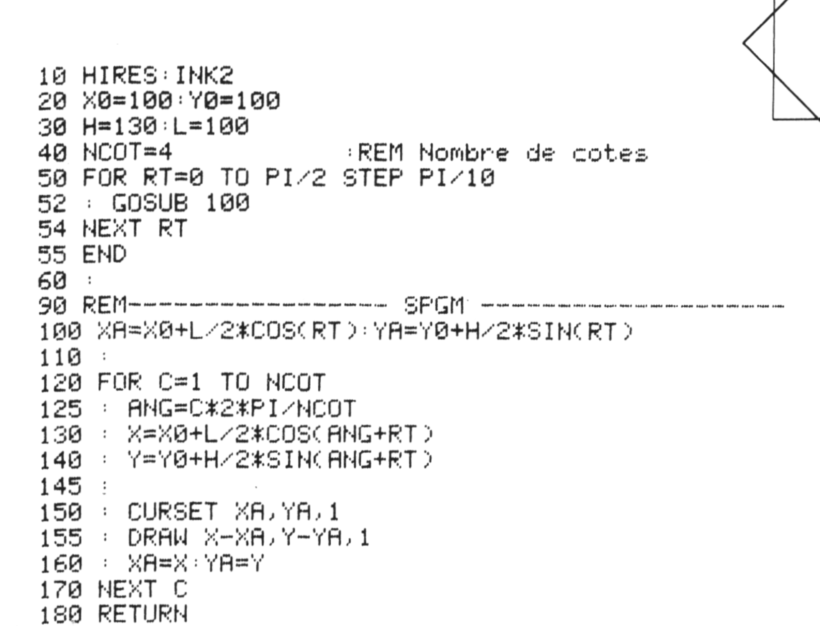
losange :

```
30 L=50:H=80
40 NCOT=4
50 GOSUB 100
51 :
52 L=L*.8:H=H*.8
54 GOSUB 100
```



Rotation d'un carré :

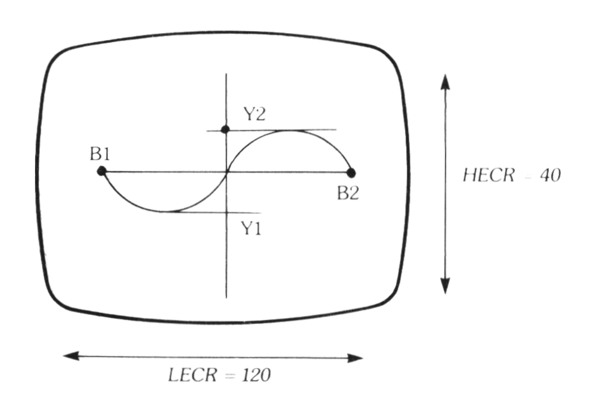
Dessine un carré qui pivote sur lui-même.




```
10 HIRES:INK2
20 X0=100:Y0=100
30 H=130:L=100
40 NCOT=4              :REM Nombre de cotes
50 FOR RT=0 TO PI/2 STEP PI/10
52 : GOSUB 100
54 NEXT RT
55 END
60 :
90 REM---------------- SPGM --------------------
100 XA=X0+L/2*COS(RT):YA=Y0+H/2*SIN(RT)
110 :
120 FOR C=1 TO NCOT
125 : ANG=C*2*PI/NCOT
130 : X=X0+L/2*COS(ANG+RT)
140 : Y=Y0+H/2*SIN(ANG+RT)
145 :
150 : CURSET XA,YA,1
155 : DRAW X-XA,Y-YA,1
160 : XA=X:YA=Y
170 NEXT C
180 RETURN
```

Tracé de courbe sur ECRAN :

Voici un programme de tracé de courbe Y=f(X).
Nous recherchons le mini et le maxi de Y : Y1 et Y2 puis nous calculons les échelles pour X et Y de façon à utiliser au mieux la dimension de l'écran.

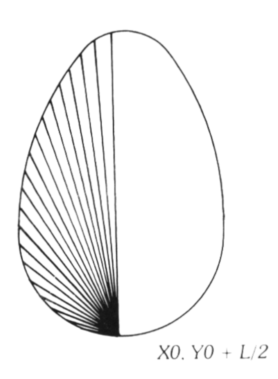



```
10 REM TRACE DE COURBE
20 :
30 HECR=150:LECR=200        :REM Hauteur/longueur ecran
40 INPUT "BORNE X1 ";B1
50 INPUT "BORNE X2 ";B2
60 INPUT "PAS       ";PAS
70 :
75 X=B1:GOSUB 290:Y1=Y:Y2=Y
80 FOR X=B1 TO B2 STEP PAS
100 : GOSUB 290
110 : IF Y<Y1 THEN Y1=Y      :REM Mini
120 : IF Y>Y2 THEN Y2=Y      :REM Maxi
130 NEXT X
135 :
140 HIRES
150 EX=LECR/(B2-B1)          :REM Echelle x
160 EY=(HECR-2)/(Y2-Y1)      :REM Echelle Y
170 REM------------------------------------ Axes ---
180 IF Y2=>0 THEN IF Y1<=0 THEN Y=HECR-ABS(Y1)*EY:CURSET 1,Y,1:DRAW
 LECR,0,1
190 IF B2=>0 THEN IF B1<=0 THEN X=ABS(B1)*EX:CURSET X,1,1:DRAW 0,HE
CR,1
200 REM------
210 FOR X=B1 TO B2 STEP PAS
220 :GOSUB 290
230 :SX=(X-B1)*EX
240 :SY=HECR-(Y-Y1)*EY
250 :CURSET SX,SY,1
260 NEXT X
265 PRINT "B1=";B1;"B2=";B2;"MINI=";Y1;"MAXI=";Y2
270 GET X$
275 TEXT:END
280 REM ------------------------ Courbe a tracer ----
290 Y=X*X-4
300 RETURN
```

Quelques dessins

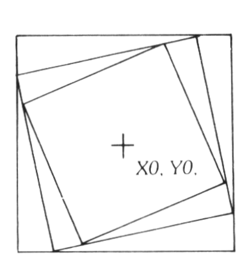
Dessine un œuf :

```
10 REM OEUF
20 :
25 X0=100:Y0=100
30 L=50
40 H=90
45 X1=X0:Y1=Y0+H
50 HIRES
55 PAPER 0:INK6
60 :
70 FOR ANG=0 TO 2*PI STEP PI/50
80 :X=X0+L*COS(ANG)
90 :Y=Y0+H*SIN(ANG)
100 :CURSET X1,Y1,1
110 :DRAW X-X1,Y-Y1,1
120 NEXT ANG
```



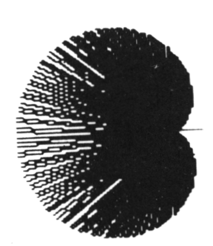
Dessine des carrés emboîtés :

```
5 REM   CARRES EMBOITES
6 :
10 HIRES:INK3
20 X=170:Y=170
30 X0=100:Y0=100
40 M=.1
60 X1=X-X0
70 Y1=Y0-Y
80 FOR I=1 TO 20
85 :CURSET X1+X0,Y0-Y1,1
90 : DRAW Y1-X1,X1+Y1,1
92 : DRAW -X1-Y1,Y1-X1,1
94 : DRAW X1-Y1,-X1-Y1,1
96 : DRAW X1+Y1,X1-Y1,1
100 :X1=X1-(X1-Y1)*M
110 :Y1=Y1-(X1+Y1)*M
130 NEXT I
```



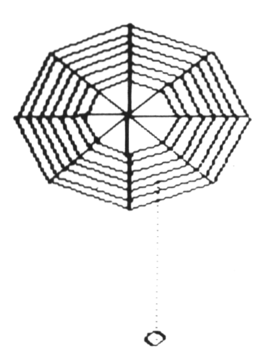
Dessine une pomme :

```
10 X0=100:Y0=100
30 HIRES
40 PAPER 7:INK 2
45 E=30
50 FOR ANG=0 TO PI*2 STEP PI/100
70 :X=E*(1-COS(ANG))*COS(ANG)
80 :Y=E*1.3*(1-COS(ANG))*SIN(ANG)
85 :IF X>-1 AND X<1 AND Y>-1 AND Y<1 THEN 100
90 :CURSET X0,Y0,1:DRAW X,Y,1
100 NEXT ANG
105 DRAW 10,0,1
110 :
120 GET X$
130 TEXT
```



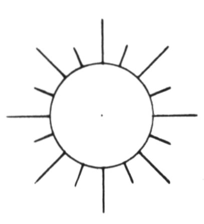
Dessine une toile... et son araignée :

```
5 REM   ARAIGNEE
6 :
10 X0=70:Y0=70
15 R=15
20 :
25 HIRES
27 PAPER6:INK0
30 FOR M=3 TO 1 STEP -.3
40 :  XA=R*M/1:YA=0
45 :  CURSET X0+R*M/1,Y0,1
50 :  FOR A=PI/4 TO 2*PI+.1 STEP PI/4
60 :    X=R*M*COS(A)
70 :    Y=R*M*SIN(A)
80 :    DRAW X-XA,Y-YA,1
85 :    IF M=3   THEN DRAW -X,-Y,1
87 :    CURSET X0+X,Y0+Y,1
88 :
90 :    XA=X : YA=Y
100 : NEXT A
120 NEXT M
130 :
140 FOR Y=100 TO 180 STEP 3
150 : CURSET 80,Y,1
160 NEXT Y
170 CIRCLE 4,1
```



Dessine un soleil :

```
5 REM          SOLEIL
6 :
10 HIRES
15 INK3
20 X0=100:Y0=100
90 REM ---------------- Centre -------
95 CURSET X0,Y0,1
100 FOR R=1 TO 30
110 : CIRCLE R,1
120 NEXT R
160 :
170 :
185 REM---------------- Rayons -------
190 FOR A=0 TO 2*PI STEP PI/20
200 : R=40+RND(1)*40
210 : X=R*COS(A):Y=R*SIN(A)
220 : CURSET X0,Y0,1:DRAW X,Y,1
230 NEXT A
```



PATTERN X :

Cette instruction permet d'effectuer des tracés en pointillés. Le dessin des pointillés est fonction de la valeur binaire de X. A chaque 1er binaire correspond un trait plein.

```
10 HIRES
20 PATTERN 121
30 CURSET 100,100,1
40 DRAW 50,0,1
```

121 → 01101111

Le programme ci-dessous dessine un cercle plein avec des alvéoles

```
10 HIRES
20 CURSET 100,100,3
30 FOR R=1 TO 30
40 : PATTERN R
50 : CIRCLE R,1
60 NEXT R
```

POINT (X,Y) :

Donne 0 si le point est en couleur BACKGROUND et − 1 si le point est en couleur FOREGROUND.

```
 10 HIRES
 20 PRINT POINT(100,100)
 30 WAIT 200
 40 CURSET 100,100,1
 50 PRINT POINT(100,100)

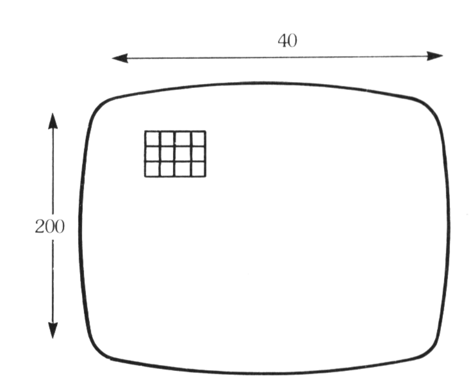
 RUN

 0
-1
```

FILL nombre de lignes, longueur, couleur :

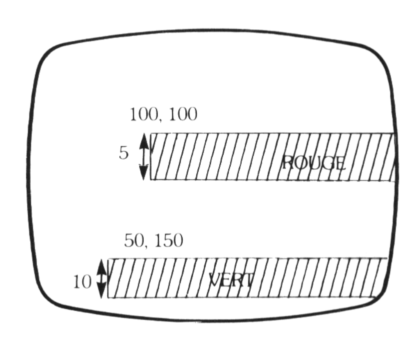
L'instruction FILL permet de modifier la couleur de fond et la couleur d'écriture de parties d'écran. L'écran est découpé en 200 × 40 cellules.

Fond : 16 < = couleur < = 23.
Ecriture : 0 < = couleur < = 7.



Le coloriage se fait à droite de la position du curseur

```
10 HIRES
20 CURSET 100,100,1
30 FILL 5,1,17
40 :
50 CURSET 50,150,1
60 FILL 10,1,18
```



La longueur spécifiée (1 sur l'exemple) n'a de sens qu'en cas de recouvrement par des FILL successifs. Dans le cas présenté, le coloriage se fait jusqu'au bout de l'écran.

```
10 REM Carres de couleur aleatoire
20 :
30 HIRES:CURSET 0,0,0:FILL 198,240/6,16
40 :
45 H=20:L=20
50 FOR N=1 TO 20
60 :X=RND(1)*190:Y=RND(1)*150:C=RND(1)*7+1
70 :CURSET X,Y,1:FILL H,L/6,16+C
80 NEXT N
```

Drapeau français :

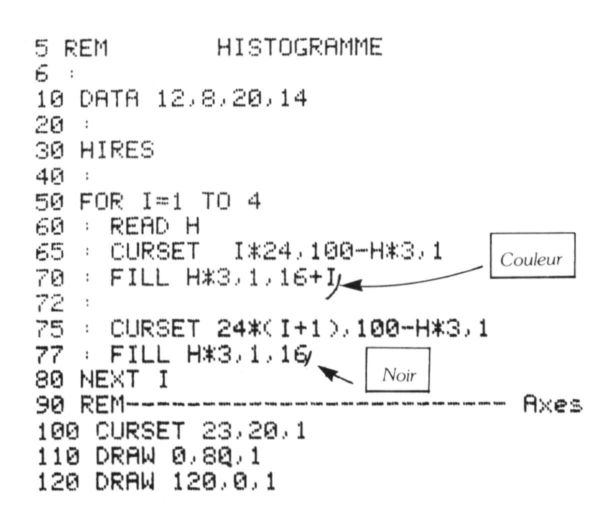
Ce programme dessine, par recouvrement de couleurs, un drapeau français.

```
5 REM     DRAPEAU FRANCAIS
6 :
10 HIRES
15 INK1
20 :
30 CURSET 48,50,1
40 FILL 60,1,20           :REM BLEU
50 :
60 CURSET 72,50,1
70 FILL 60,1,23           :REM BLANC
80 :
90 CURSET 96,50,1
100 FILL 60,1,17          :REM ROUGE
110 :
120 CURSET 120,50,1
130 FILL 60,1,16
135 :
140 CURSET 49,50,1
150 DRAW 0,100,1
160 CURSET 80,58,1
170 X$="FRANCE"
180 FOR I=1 TO LEN(X$)
190 CHAR ASC(MID$(X$,I,1)),0,1
195 CURMOV 0,8,0
200 NEXT I
```

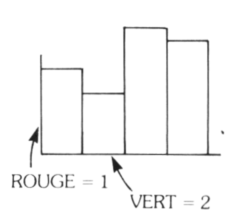
FRANCE

60
1) BLEU
72
2) BLANC
96
3) ROUGE
120
4) NOIR

Histogramme :



```
5 REM       HISTOGRAMME
6 :
10 DATA 12,8,20,14
20 :
30 HIRES
40 :
50 FOR I=1 TO 4
60 : READ H
65 : CURSET  I*24,100-H*3,1
70 : FILL H*3,1,16+I
72 :
75 : CURSET 24*(I+1),100-H*3,1
77 : FILL H*3,1,16
80 NEXT I
90 REM------------------------- Axes
100 CURSET 23,20,1
110 DRAW 0,80,1
120 DRAW 120,0,1
```

Rectangles multicolores :

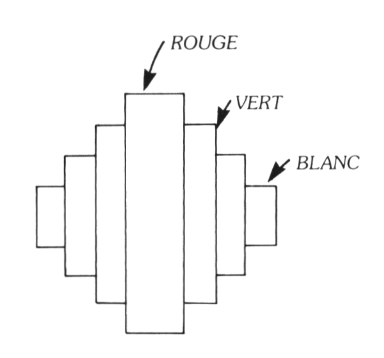
• Nous dessinons un rectangle rouge puis nous superposons un rectangle vert. Essayez le programme sans l'instruction 20.

```
5 HIRES
7 :
20 FILL 200,39,16
35 REM------ Rectangle rouge
40 CURSET 40,20,1
50 FILL 30,10,17
55 REM------ Rectangle vert
60 CURSET 50,25,1
70 FILL 6,5,18
80 :
```

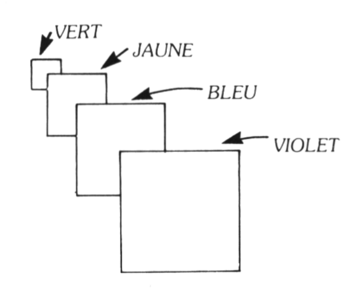
VERT
ROUGE

• Des tranches de couleurs sont superposées.

```
5 HIRES
7 :
20 FILL 200,39,16
35 :
80 :
100 FOR C=6 TO 1 STEP-1
110 : CURSET(12-C)*12,C*8,1
120 : FILL (10-C)*16,C*4,16+C
125 : WAIT 100
130 NEXT C
```




```
10 HIRES
20 :
25 FILL 150,39,16
30 FOR F=2 TO 7
35 : X=F*6*2
37 : Y=F*8
40 : CURSET X,Y,1
50 :
60 : FILL F*8 ,F*2,16+F
70 NEXT F
```



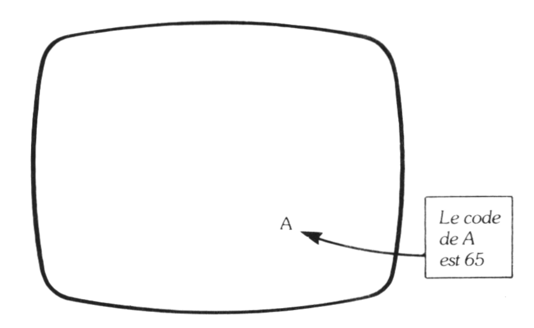
CHAR Code, jeu caractères, type couleur :

Cette instruction permet d'afficher un caractère en haute résolution. Le code doit être compris entre 32 et 127.

Code : 32 < code < 127

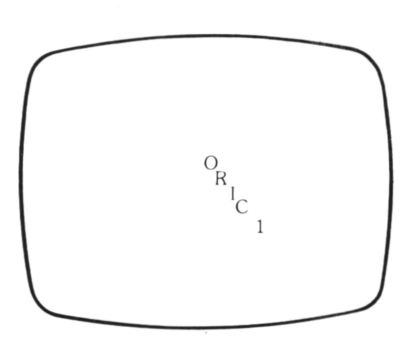
Jeu de caractères 0 ou 1 (jeu normal ou alterné)
Pour afficher la lettre A au point de coordonnées 100, 100

```
10 HIRES
20 CURSET 100,100,3
30 CHAR 65,0,1
```



Pour afficher une chaîne, il faut décomposer celle-ci en caractères et afficher chacun d'eux.

```
10 HIRES
20 N$="ORIC1"
30 :
40 CURSET 100,100,3
50 FOR P=1 TO LEN(N$)
60 :C=ASC(MID$(N$,P,1))
70 :CHAR C,0,1
80 :CURMOV 10,10,3
90 NEXT P
```



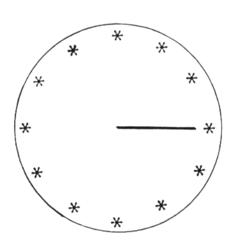
Histogramme circulaire :

Remarque : Pour afficher les pourcentages à l'intérieur de l'histogramme, il faut convertir les pourcentages en chaîne (par STR$) puis afficher le résultat caractère par caractère. Le premier caractère de la chaîne qui devrait être un espace ne sera pas affiché (son code est 2).

```
1 REM     HISTOGRAMME CIRCULAIRE
2 :
5 H(1)=.2:H$(1)="TARTE"
6 H(2)=.3:H$(2)="CROISSANT"
7 H(3)=.1:H$(3)="BRIOCHE"
8 H(4)=.3:H$(4)="ECLAIR"
9 H(5)=.1:H$(5)="PAIN"
10 :
15 HIRES
17 INK2
20 X0=120:Y0=100
40 R=30
45 CURSET X0,Y0,1:CIRCLE R,1
46 REM----------------------------------------------
47 :
48 AA=0
50 FOR P=1 TO 5
60 : A=AA+(PI*2)*H(P)
62 : CURSET X0,Y0,C
65 : X=R*COS(A):Y=R*SIN(A)
67 : DRAW X,Y,1                    :REM Tranche
70 :
72 : AT=AA+PI*H(P)                 :REM Position texte
73 : X$=H$(P)
74 : X=R*1.3*COS(AT):Y=R*1.3*SIN(AT)
75 : IF AT>PI/2 AND AT<3*PI/2 THEN X=X-LEN(X$)*6
78 :
80 : CURSET X0+X,Y0+Y,0            :REM Texte
82 : FOR I=1 TO LEN(X$)
84 :   CHAR ASC(MID$(X$,I,1)),0,1
86 : CURMOV 6,0,0
88 : NEXT I
90 :
98 : AA=A
100 NEXT P
```

TARTE
CROISSANT
BRIOCHE

Horloge :



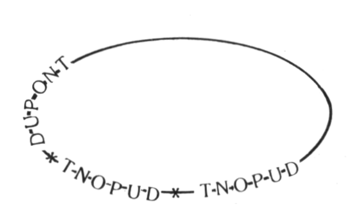
```
10 REM     HORLOGE
20 :
30 HIRES:INK1
40 X0=100:Y0=100:R=50
50 :
60 REM---------------------Dessin horloge ----
70 CURSET X0,Y0,1:CIRCLE R,1
80 FOR A=0 TO 2*PI STEP PI/6
90 : X=R*.8*COS(A):Y=R*.8*SIN(A)
100 : IF A>PI/2 AND A<3*PI/2 THEN X=X-5
110 : CURSET X0+X,Y0+Y,0
120 : CHAR 42,0,1
130 NEXT A
140 REM------------------- Mouvement --------
150 R=R*.7
160 FOR A=0 TO 1000 STEP PI/30
170 : IF A>2*PI THEN 310
180 : CURSET X0,Y0,1
190 : IF X1<>0 OR Y1<>0 THEN DRAW X1,Y1,2
200 : CURSET X0,Y0,1
210 : X=R*COS(A):Y=R*SIN(A)
220 : DRAW X,Y,1
230 : SOUND 1,4,14
240 : WAIT 4
250 : SOUND 1,4,0
260 : WAIT 40
270 :
280 : X1=X:Y1=Y
290 NEXT A
300 REM------------------- Explosion --------
310 EXPLODE
320HIRES:INK1
330 CURSET X0,Y0,1
340 FOR R=2 TO 30:CIRCLE R,1:NEXT R
```

Remarque : On peut modifier l'instruction WAIT 40 (ligne 260) de façon à ce qu'un tour de cadran corresponde à une minute.

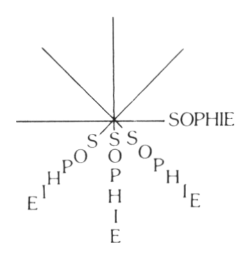
Guirlandes circulaires :

Votre nom apparaît sous forme d'une guirlande circulaire.

```
5 REM      GUIRLANDE CIRCULAIRE
6 :
10 HIRES:INK6
20 INPUT "VOTRE NOM  ";N$
30 N$=N$+"*"+N$:IF LEN(N$)<50 GOTO 30
40 :
45 CLS
50 FOR P=1 TO 40
60 :    A=P*PI/20
70 :    Y=SIN(A)*80+100
75 :    X=COS(A)*100+120
80 :    X$=MID$(N$,P,1)
90 :    CURSET X,Y,0
95 :    CHAR ASC(X$),0,1
100 NEXT P
```




```
5 REM      GUIRLANDE
6 :
10 HIRES:INK6
20 INPUT "VOTRE NOM  ";N$
25 N$=LEFT$(N$,8)
30 :
40 :
45 CLS
50 FOR P=1 TO 21
60 :   A=P*PI/10
65 :   FOR R=1 TO LEN(N$)
70 :    Y=SIN(A)*R*10+80
75 :    X=COS(A)*R*12+120
80 :    X$=MID$(N$,R,1)
90 :    CURSET X,Y,0
95 :    CHAR ASC(X$),0,1
97 :   NEXT R
100 NEXT P
```



CARACTÈRES SPÉCIAUX

Des caractères particuliers permettent (en mode TEXT) de changer la couleur des caractères, la couleur de fond où sont écrits les caractères, de doubler la taille des caractères, etc.

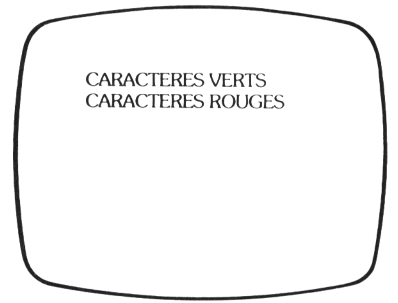
Couleur des caractères :

La couleur des caractères se choisit par :

PRINT CHR$(27) ; «@ à G» ; «MESSAGE» ;

Le programme ci-dessous affiche une ligne de caractères verts et une ligne de caractères rouges.

```
10 CLS
20 E$=CHR$(27)
30 PRINT " ";E$;"B";"CARACTERES VERTS";
40 PRINT     E$;"A";"CARACTERES ROUGES"
```



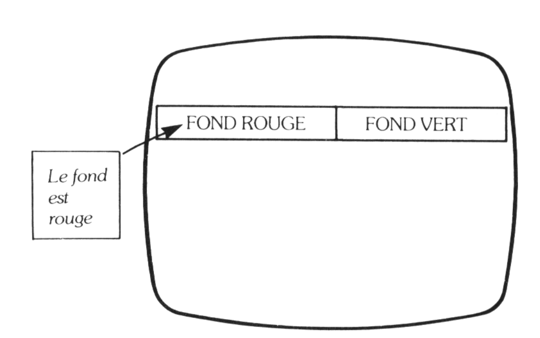
Remarque : Chaque saut de ligne annule le changement de couleur. Par conséquent, l'instruction PRINT CHR$(27) ; «@ à G» doit être exécutée pour chaque ligne.

Couleur de fond :

De la même façon, la couleur de fond (où sont écrits les caractères) se choisit par :

PRINT CHR$(27) ; «P à W» ;

```
10 CLS
20 E$=CHR$(27)
30 PRINT " ";E$;"Q";"FOND ROUGE";
40 PRINT     E$;"R";"FOND VERT"
```



Ci-dessous, pour 4 couleurs de fond différentes, nous affichons le même message avec 4 couleurs de caractères différentes.

Essayez le même programme avec un point virgule à la fin de 80 (PRINT « BLABLA » ;) et 95 PRINT. Vous obtiendrez 4 couleurs de caractères sur la même ligne.

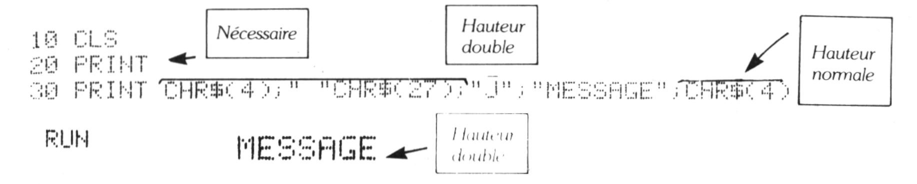
```
10 CLS
20 E$=CHR$(27)
30 :
40 FOR CBK=0 TO 3
50 :  FOR CFR=0 TO 3
60 :    PRINT " ";E$;CHR$(80+CBK);  :REM Background
70 :    PRINT E$;CHR$(64+CFR);      :REM Foreground
80 :    PRINT "BLABLA"
90 :  NEXT CFR
100 NEXT CBK
```


Couleurs avec PLOT :

Les couleurs d'écriture et de fond peuvent également être choisies avec PLOT X, Y, CHR$ (couleur). Couleur est compris entre 0 et 7 pour l'écriture et 16 et 23 pour la couleur de fond.

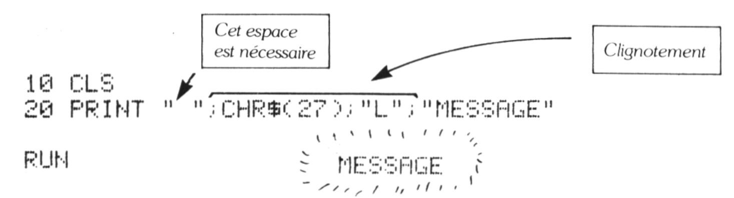
```
5 REM   Couleurs avec PLOT
6 REM
10 CLS:PAPER7:INK4
20 X$=CHR$(2)+"VERT":PLOT 10,10,X$
30 :
40 Y$=CHR$(16+1)+"FOND ROUGE"+CHR$(16+7):PLOT 10,12,Y$
```

CARACTERES DE TAILLE DOUBLE : CHR$(27); «J»

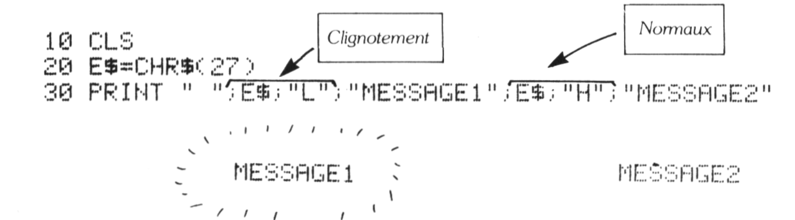


Remarque : L'instruction PRINT en ligne 20 est nécessaire ; elle permet de placer le curseur sur une ligne paire.

Caractères clignotants : CHR$(27); «L»



Retour aux caractères normaux : CHR$(27); «H»



Utilisation de PLOT :

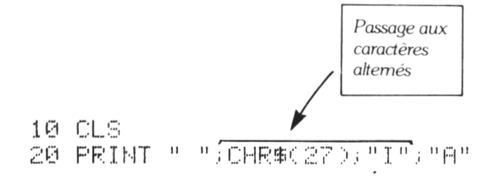
CHR$ (10) → Caractères de taille double.
CHR$ (12) → Caractères clignotants
CHR$ (8) → Caractères normaux.

```
5 REM Caracteres de taille double
6 REM
10 CLS:X$=CHR$(10)+"TAILLE DOUBLE"
20 PLOT 10,11,X$
30 PLOT 10,12,X$
40 :
50 REM Caracteres clignotants
60 :
70 Y$=CHR$(12)+"CLIGNOTANT"+CHR$(8)
80 PLOT 10,20,Y$
```

CARACTERES ALTERNES :

Il existe un jeu de caractères alternés (ou semi-graphiques) utiles pour la composition de dessins.

On y accède par : **CHR$(27) ; « I »**



Le programme ci-dessous affiche tous les caractères alternés.

```
10 CLS
20 FOR C=32 TO 127
30 :PRINT C,CHR$(27);"I";CHR$(C)
40 :PRINT
50 :WAIT 50
60 NEXT C
```

Cet autre programme affiche les caractères alternés sur 3 colonnes et en couleur :

```
5 REM    CARACTERES SEMI-GRAPHIQUES
6 :
10 CLS
20 E$=CHR$(27)                          :REM Escape
30 :
40 FOR C=34 TO 99
42 : PRINT " ";
45 : PRINT E$;"A";                      :REM Rouge
50 : PRINT E$;"H";C;CHR$(C);            :REM Normal
60 :
70 : PRINT E$;"I";E$;"D";CHR$(C);       :REM Alterne/bleu
90 : Q=INT(C/3):R=C-Q*3:IF R=0 THEN PRINT
100 NEXT C
```

34 " ■   35 ≠ ■   36 $ ■

37 % ■   38 &   39 '

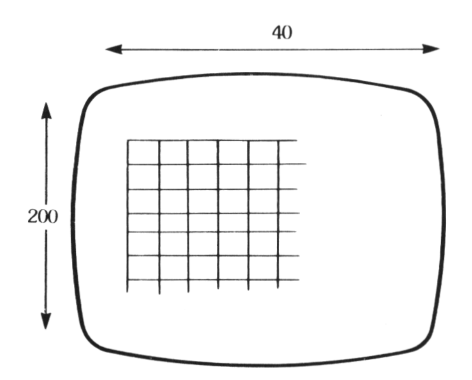
Nous vous proposons, avec ce programme, de déplacer un avion de gauche à droite sur l'écran.

```
10 REM     AVION
20 :
110 AV$=CHR$(44)+CHR$(100)+CHR$(36)
130 :
135 LORES 1
140 FOR X=1TO 30
145 : PLOT X-1,15," "   :REM Effacement arriere avion
150 : PLOT X,15,AV$
155 : WAIT 5
160 NEXT X
170 :
180 GET X$
200 TEXT
```

FONCTIONS PARTICULIÈRES

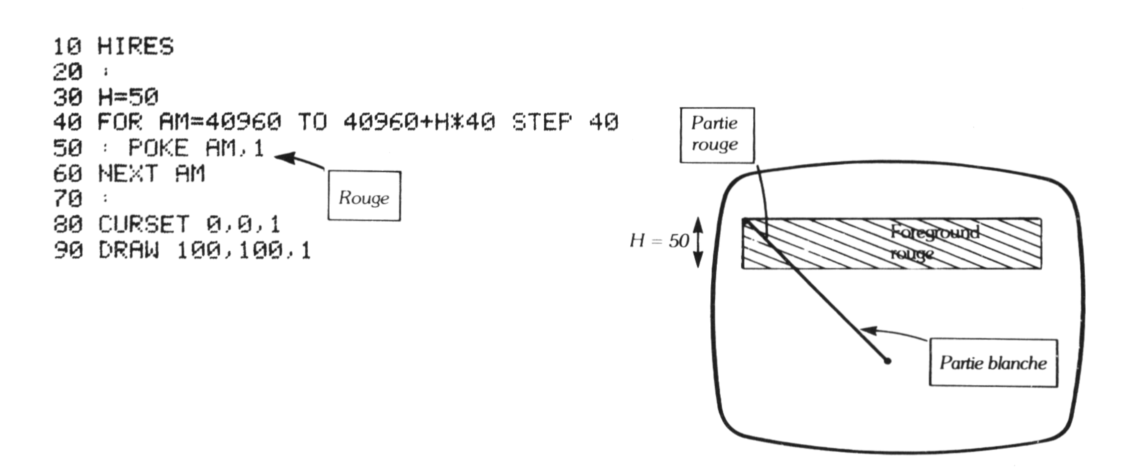
Choix d'une couleur pour une partie d'écran :

En mode graphique, il n'existe pas d'instruction BASIC pour choisir les couleurs FOREGROUND et BACKGROUND (de fond) pour une partie d'écran.
Il faut recourir à l'instruction POKE. L'écran est divisé en 200 * 40 cellules.



POKE adresse mémoire, couleur FOREGROUND

Pour définir une bande rouge de hauteur H, nous écrivons :
Naturellement, la bande FOREGROUND rouge ne se voit pas à l'écran.




```
10 HIRES
20 :
30 H=50
40 FOR AM=40960 TO 40960+H*40 STEP 40
50 : POKE AM,1
60 NEXT AM
70 :
80 CURSET 0,0,1
90 DRAW 100,100,1
```

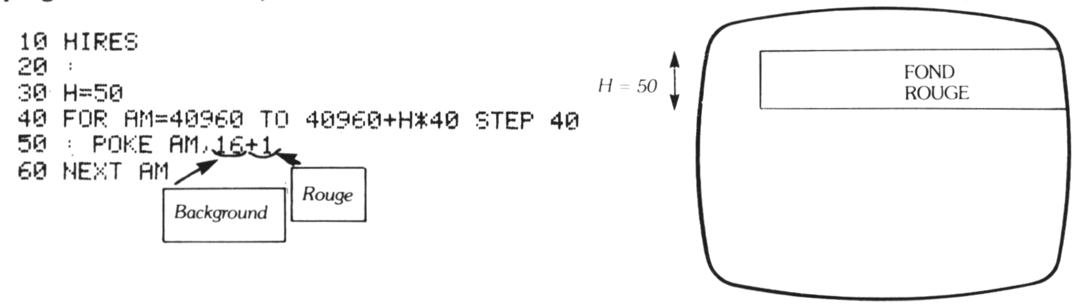
Ci-dessous, nous traçons une bande située au milieu de l'écran,

```
10 HIRES
20 :
30 X1=20:Y1=100:H=50
40 FOR AM=40960+Y1*40 TO 40960+(Y1+H)*40 STEP 40
50 : POKE AM+X1,2
60 NEXT AM
70 :
80 CURSET 50,110,1
90 DRAW 100,0,1
```

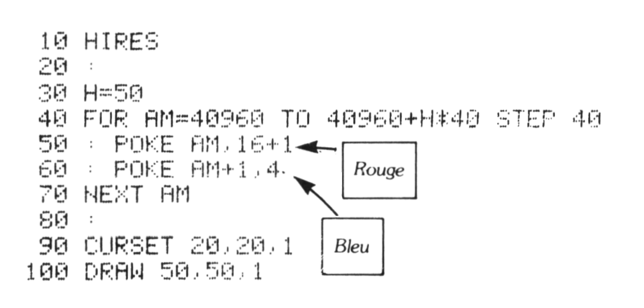
Vert

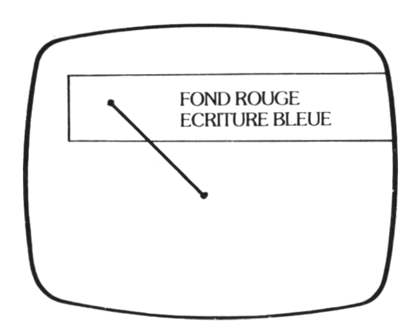
POKE adresse mémoire, couleur fond

De la même façon, la couleur de fond (BACKGROUND) d'une partie d'écran peut être programmée. Il faut ajouter 16 au numéro de la couleur.



La séquence ci-dessous définit à la fois une partie FOREGROUND bleu et une partie BACKGROUND rouge.





Droites multicolores :

```
2 REM----- TRACE DE DROITES DE COULEURS DIFFERENTES
3 :
5 HIRES
10 B=40960
20 REM------------------- DROITE VERTE
25 C=2
30 X0=80 :Y0=50
32 DX=+40:DY=60
34 GOSUB 60
35 REM------------------- DROITE ROUGE
36 X0=70:Y0=20
37 DX=+50:DY=10
38 C=1:GOSUB 60
50 :
55 STOP
59 REM--------------- SOUS PROGRAMME
60 X1=INT(X0/6)
70 Y1=Y0
75 D1=INT(DX/6):D2=DY
80 :
100 DIST=SQR(D1*D1+D2*D2)
110 CX=D1/DIST:CY=D2/DIST
120 :
130 FOR D=1 TO DIST+1
140 : X=INT(X1+D*CX)-1
150 : Y=INT(Y1+D*CY)
155 : AM=B+(40*Y)+X
160 : POKE AM,C
170 NEXT D
180 :
200 CURSET X0,Y0,1
210 DRAW DX,DY,1
220 RETURN
```

FILL nombre de lignes, longueur, couleur foreground :

- La couleur foreground d'une partie d'écran peut également être choisie avec FILL.

```
5 REM  Couleur foreground avec FILL
6 REM
10 HIRES:X1=20:Y1=30:H=30:L=100
20 CURSET X1,Y1,3:FILL H,1,2
30 CURSET X1+L,Y1,3:FILL H,1,7
40 CURSET 0,Y1+10,3:DRAW 150,0,1
```

Feux bicolores :

Nous dessinons 2 cercles pleins. En choisissant les couleurs FOREGROUND, nous les faisons apparaître en rouge et vert alternativement.

```
5 REM      FEUX BICOLORES
6 :
10 HIRES
12 H=30              :REM HAUTEUR TRANCHE
15 :
20 C1=1:C2=2:GOSUB 170   :REM COULEURS
30 GOSUB 100             :REM CERCLES
35 WAIT 200
40 C1=2:C2=1:GOSUB 170   :REM COULEURS
45 WAIT 200:GOTO 20
50 END
70 REM------------- TRACE CERCLES ----
100 CURSET 100,H/2,1
110 FOR R=1 TO 10
120 : CIRCLE R,1
130 NEXT R
135 :
140 CURSET 100,H+H/2,1
145 FOR R=1 TO 10
147 :CIRCLE R,1
150 NEXT R
160 REM------------ SPGM COULEURS ---
170 FOR AM=40960 TO 40960+40*H STEP 40
180 : POKE AM,C1
190 : POKE AM+40*H,C2
200 NEXT AM
210 RETURN
```

30
30
Foreground rouge
Foreground vert

En changeant les couleurs FOREGROUND, le cercle rouge devient vert et le cercle vert devient rouge.

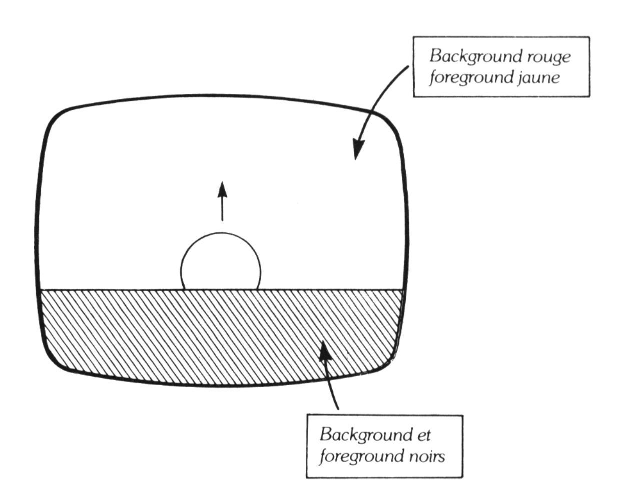
Le sous-programme en 160 peut être remplacé par :

```
170 CURSET 0,0,3
180 FILL H,1,C1
190 FILL H,1,C2
200 RETURN
```

Soleil levant :

Un cercle jaune plein se déplace du bas de l'écran vers le haut. La partie basse de l'écran ayant une couleur FOREGROUND identique à la couleur BACKGROUND, le tracé du cercle n'apparaît pas dans cette partie.

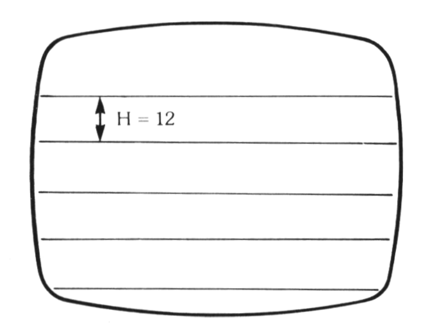
```
10 REM    SOLEIL LEVANT
15 :
20 HIRES
60 REM -------------- Fond haut ecran rouge/ecriture rouge
70 CURSET 0,0,3:FILL 135,1,3       :REM jaune
80 CURSET 10,0,3:FILL 135,1,16+1   :REM rouge
91 REM---------------- Fond bas ecran noir ------
92 CURSET 0,135,3:FILL 60,1,0
98 REM---------------------------------------------
100 X0=100:Y0=100
110 :
112 FOR D=65 TO 1 STEP -1  :REM Deplacement
115 :CURSET X0,Y0+D,1
130 :CIRCLE 29,1
145 :
150 :CIRCLE 31,0
200 NEXT D
```

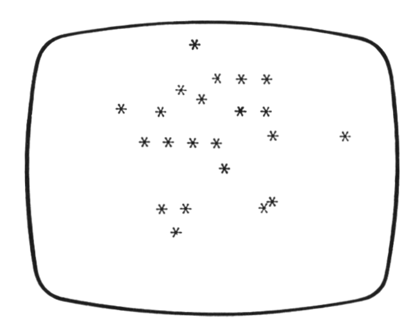


Ciel étoilé multicolore :

Des tranches de couleur FOREGROUND (hauteur 12) sont choisies au hasard. Des « * » sont affichées au hasard sur l'écran. Leur couleur correspond à celle de la tranche où ils sont affichés.

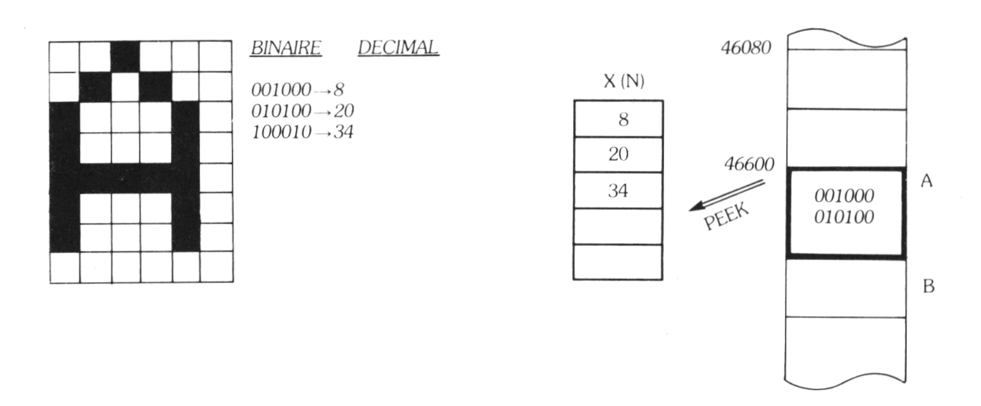
```
5 REM       CIEL ETOILE  MULTICOLORE
6 :
10 HIRES
15 HT=12            :REM Hauteur tranche culeur
18 REM-------------- Foreground couleurs aleatoires
20 FOR AM=40960 TO 40960+200*40 STEP HT*40
30 : C=RND(1)*7+1
35 : FOR H=0 TO HT-1
36 :   POKE AM+H*40,C
37 : NEXT H
38 :
40 NEXT AM
50 REM------------------------- Etoiles -----
60 FOR E=1 TO 50
70 : X=RND(1)*230:Y=RND(1)*180
80 : CURSET X,Y,0
90 : CHAR 42,0,1
100 NEXT E
```





ACCES AUX CODES ECRAN DES CARACTERES :

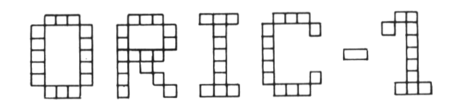
Chaque caractère affiché à l'écran est un ensemble de points allumés à l'intérieur d'une matrice 8 × 8.
8 octets en mémoire sont utilisés pour représenter cette matrice.



Si nous connaissons l'adresse de la zone mémoire où est stockée la définition des caractères écran, nous pourrons y accéder grâce à l'instruction « PEEK adresse mémoire » et « grossir » ces caractères à l'écran en utilisant la basse résolution.

Les instructions 1000 à 1100 convertissent en binaire les valeurs décimales (obtenues par PEEK) en valeurs binaires.

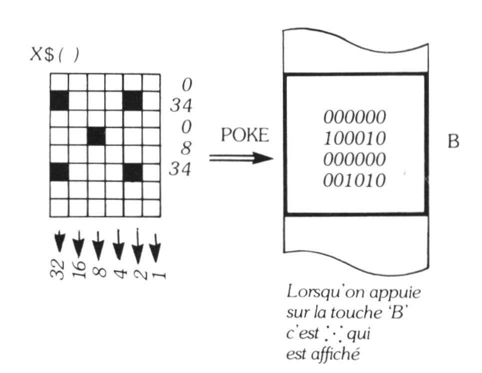
ORIC-1 s'affiche en gros caractères, la couleur de chaque «pavé» est choisie au hasard.



```
5 REM       GROS CARACTERES MULTICOLORES
6 :
10 CLS
15 LORES0
20 N$="ORIC1"
30 :
40 FOR P=1 TO LEN(N$)
50 :C$=MID$(N$,P,1)
60 :C=ASC(C$):A=46080:D=C*8
70 :FOR N=0 TO 7
80 :  X(N)=PEEK(A+D+N)
90 :NEXT N
100 GOSUB 1000
190 NEXT P
200 END
1000 REM---------------- Dessine 1caractere
1010 FOR N=0 TO 7                  :REM 8 LIGNES
1030 : FOR M=7 TO 0 STEP-1         :REM CONVERSION BINAIRE
1040 :   Q=INT(X(N)/2):R=X(N)-(Q*2)
1050 :   X(N)=Q
1070 :   IF R=0 THEN X=16 ELSE X=17+RND(1)*4
1080 :   PLOT (P-1)*7+M,10+N,X
1090 : NEXT M
1100 NEXT N
1110 RETURN
```

MODIFICATION D'UN CARACTERE D'ECRAN :

Avec le programme précédent, nous accédions aux codages internes des caractères affichés à l'écran. Ici, nous modifions ce codage de façon à changer l'affichage du caractère « B » qui devient



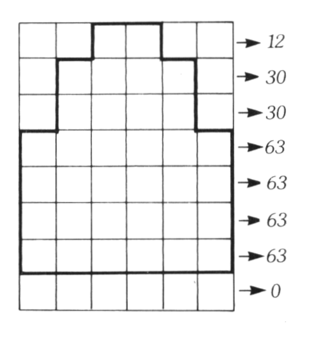
Pour cela, nous codons en DATA le dessin du caractère (ici un DE). Nous convertissons le codage binaire en décimal puis nous le rangeons par :

POKE adresse mémoire, code décimal.

Les valeurs décimales peuvent être obtenues en ajoutant 2055 PRINT ND.

Les instructions 1000 à 1110 ne sont pas nécessaires.

Autre exemple :
Pour dessiner une silhouette de soucoupe volante, nous faisons :




```
10 DATA 12,30,30,63,63,63,63,0
20 A=46080:C$="!":C=ASC(C$):D=C*8
25 :
30 FOR N=0 TO 7:READ ND:POKE A+D+N,ND:NEXT N
60 :
70 CLS:PLOT 10,10,C$
80 PLOT 15,15,CHR$(2)+C$            :REM soucoupe vert
```

```
2 REM  DESSIN D'UN DE
3 :
5 CLS
7 C$="B"                ' Caractere a changer
8 C=ASC(C$)
10 A=46080:D=C*8        ' Adresse memoire ecran
12 LORES 0
13 REM---------------- Dessin caractere
14 X$(0)="       "
15 X$(1)="1    1 "
16 X$(2)="       "
17 X$(3)="  1    "
18 X$(4)="       "
19 X$(5)="1    1 "
20 X$(6)="       "
22 X$(7)="       "
30 :
100 GOSUB 1000
110 GOSUB 2000
115 PRINT C$
200 END
1000 REM---------- Affichage gros caracteres
1010 FOR N=0 TO 7
1020 : FOR M=1 TO LEN(X$(N))
1030 : R=VAL(MID$(X$(N),M,1))
1040 :  R=VAL(MID$(X$(N),M,1))
1050 :  IF R=1 THEN X=20 ELSE X=16
1060 :  PLOT M,10+N,X
1090 :NEXT M
1100 NEXT N
1110 RETURN
1990 REM--------- Conversion binaire-->decimal
2000 FOR N=0 TO 7
2010 : ND=0
2020 : L=LEN(X$(N))
2030 : FOR M=1 TO L
2040 :  R=VAL(MID$(X$(N),M,1))
2050 :  ND=ND+R*(2^(L-M))
2060 : NEXT M
2070 : POKE (A+D+N),ND       :REM Modification memoire ecran
2075 PRINT ND
2080 NEXT N
2090 RETURN
2100 REM===========================
2105 REM      METHODE DIRECTE
2107 REM  FAIRE RUN 2110
2110 A=46080:C$="B":C=ASC(C$):D=C*8
2120 DATA 0,34,0,8,0,34,0,0
2130 :
2140 FOR N=0 TO 7
2150 :READ ND
2160 :POKE A+D+N,ND
2170 NEXT N
```

Mini-interpréteur de boucles :

Pour expliquer comment fonctionne un 'Interpréteur Basic', nous avons écrit en DATA un pseudo-programme.

```
30 REM ----------------------------------- pseudo programme interprete
40 DATA REPETE 2
50 DATA   PRINT "Grande boucle"
60 DATA   REPETE 3
70 DATA     PRINT "   Boucle interne"
80 DATA   ENCORE
90 DATA ENCORE
100 DATA *
110 REM---------------------------------------------------
120 FOR I=1 TO 100
130  READ X$:IF X$="*" THEN NC=I-1:GOTO 170

140  CD$(I)=X$
150 NEXT I
160 :
170 PL=1           :REM pointeur de ligne courante
180 :
190 LG$=CD$(PL)
200 C$=LEFT$(LG$,4)
210 IF C$="REPE" THEN GOTO 330
220 :
230 IF C$="ENCO" THEN GOTO 370
240 :
250 PRINT PL;LG$
260 :
270 PL=PL+1
280 IF PL>NC THEN STOP
290 GOTO 190
300 REM ----------------------------------------------------------- Repete
310 REM      PR --> pointeur pile RP() contenant no ligne apres REPETE
320 REM                           NB() contenant nb de boucles
330 FOR P=1 TO LEN(LG$)
332  IF MID$(LG$,P,1)=" " THEN 340
334 NEXT P
336 P=0
340 PR=PR+1:RP(PR)=PL+1:NB(PR)=VAL(RIGHT$(LG$,LEN(LG$)-P))
350 PL=PL+1:GOTO 190
360 REM----------------------------------------------------------- Encore
370 NB(PR)=NB(PR)-1:IF NB(PR)>0 THEN PL=RP(PR):GOTO 190
380 PR=PR-1:PL=PL+1:GOTO 190
```

Table CD$( )

1	REPETE 2
2	PRINT "GRANDE BOUCLE"
3	REPETE 3
4	PRINT "BOUCLE INTERNE"
5	ENCORE
6	ENCORE

PL → (ligne 3)

PR →
2

PILE RP( )
N° ligne

2

PILE NB( )
nombre de boucles

```
RUN

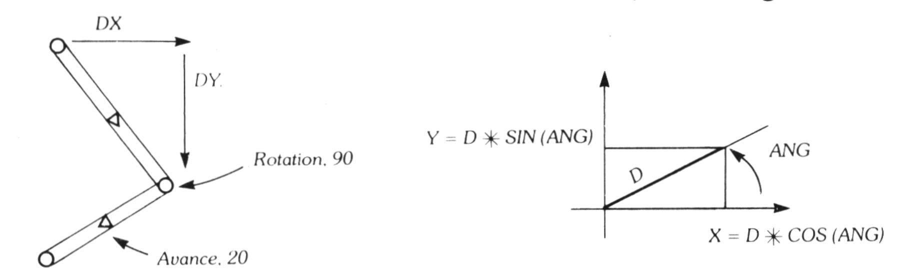
Grande boucle
  Boucle interne
  Boucle interne
  Boucle interne
Grande boucle
  Boucle interne
  Boucle interne
  Boucle interne
```

Tracé de figures :

La plupart des langages proposent des ordres graphiques où sont spécifiées les coordonnées X, Y des droites à tracer. Le langage LOGO propose des ordres graphiques originaux. Nous en avons simulé deux en BASIC :

AVANCE distance : Trace d'une droite d'une longueur égale à 'distance'.

ROTATION angle : Change la direction du tracé en lui ajoutant 'angle'.




```
10 REM --- Trace de dessins ---
20 HIRES:INK 3
30 X0=100:Y0=100
35 :
40 DATA ROTATION,45
50 DATA AVANCE,50
60 DATA ROTATION,90
70 DATA AVANCE,50
72 DATA ROTATION,90
74 DATA AVANCE,50
76 DATA ROTATION,90
78 DATA AVANCE,50
80 DATA FIN
90 :
140 READ CM$
150 :
160 IF CM$="FIN" THEN 350
170 IF CM$="ROTATION" THEN READ R:ANG=ANG+R:GOSUB 210
180 IF CM$="AVANCE" THEN READ DIST:GOSUB 290
190 GOTO 140
200 REM ----------------Angle -----------------------
210 IF ANG=>360 THEN ANG=ANG-360
220 AR=ANG/360*2*PI
230 CX=COS(AR):CY=SIN(AR)
250 IF CX>-.01 AND CX<.01 THEN CX=0
260 IF CY>-.01 AND CY<.01 THEN CY=0
270 RETURN
280 REM --------------- Trace droite ---------------
290 :
300 DX=DIST*CX:DY=DIST*CY*1.3: REM Deplacement X,Y
310 CURSET X0,Y0,1:DRAW DX,DY,1
320 X0=X0+DX:Y0=Y0+DY
330 RETURN
340 :
350 GET A$
360 TEXT
```

Mini-interpréteur LOGO

Nous interprétons en BASIC quelques ordres graphiques du langage LOGO.

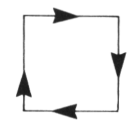
Instructions de base :

DISTANCE 10	La distance courante devient égale à 10.
AVANCE	Avance de la distance courante.
AVANCE 10	Avance d'une distance de 10.
ROTATION 90	La direction courante est modifiée de 90 degrés.
DIST. + 3	La distance courante est augmentée de 3.
LEVE	Le tracé n'a plus lieu.
BAISSE	Le tracé reprend.

Boucles :

Les instructions entre REPETE X et ENCORE sont exécutées X fois.

```
REPETE 4
 ROTATION 90
 AVANCE
ENCORE
```

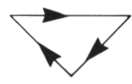


Cette séquence dessine un carré.

Fonctions :

Des 'fonctions' se déclarent ainsi.

```
DEFI TRIANGLE
REPETE 3
 AVANCE
 ROTATION 120
ENCORE
FINI
```



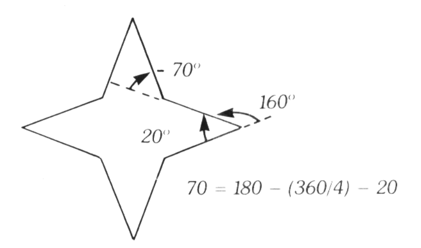
Pour dessiner un triangle, il suffit ensuite d'écrire :

```
TRIANGLE
```

Au moment de la lecture des instructions, la 'fonction' est remplacée par la suite des instructions qui y sont écrites.

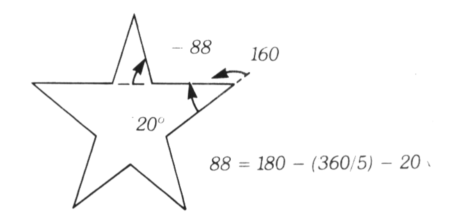
Dessin d'une étoile 4 branches :

```
REPETE 4
 AVANCE 10
 ROTATION 160
 AVANCE 10
 ROTATION -70
ENCORE
```



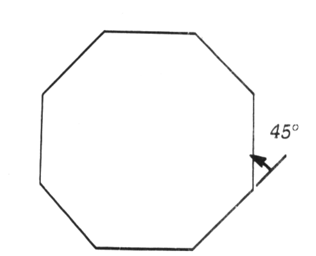
Etoile 5 branches :

```
REPETE 5
 AVANCE 10
 ROTATION 160
 AVANCE 10
 ROTATION -88
ENCORE
```



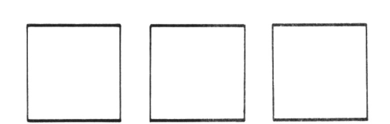
Octogone :

```
REPETE 8
 AVANCE 10
 ROTATION 45
ENCORE
```



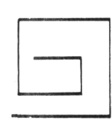
3 carrés :

```
REPETE 3
 DISTANCE 5
 BAISSE
 CARRE
 LEVE
 AVANCE 15
ENCORE
```



Spirale :

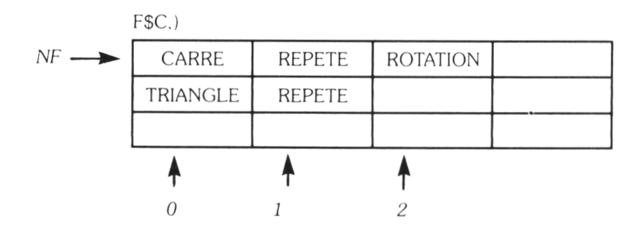
```
DISTANCE 5
REPETE 6
 AVANCE
 ROTATION 90
 DIS+ 2
ENCORE
```



```
10 REM ----MINI SIMULATEUR LOGO----
20 :
30 HIRES:INK3
40 :
50 DIM CD$(50)
60 X0=100:Y0=100:CX=1:CY=0:DIST=20
70 REM ---- Pseudo Programme interprete -----
80 :
90 DATA DEFI CARRE
100 DATA REPETE 4
110 DATA  ROTATION 90
120 DATA  AVANCE
130 DATA ENCORE
140 DATA FINI
150 :
160 DATA DEFI TRIANGLE
170 DATA REPETE 3
180 DATA  ROTATION 120
190 DATA  AVANCE
200 DATA ENCORE
210 DATA FINI
220 :
230 DATA DISTANCE 30
240 DATA CARRE
250 :
260 DATA LEVE
270 DATA AVANCE 40
280 DATA BAISSE
290 DATA AVANCE 50
300 DATA TRIANGLE
510 DATA *
520 REM ----------- Lecture des commandes dans CD$() --
530 I=1
540 :
550 READ X$ : PRINT X$
560 :
570 IF LEFT$(X$,4)="DEFI" THEN GOSUB 1250 : GOTO 550
580 :
590 IF X$="*" THEN NC=I : GOTO 670
600 FOR K=1 TO 5
610 IF X$=F$(K,0) THEN GOSUB 1330
620 NEXT K
630 :
640 CD$(I)=X$
650 I=I+1 : GOTO 550
```

```
660 REM-----------
670 PL=1 : REM -- Pointeur de ligne courante --
680 :
690 LG$=CD$(PL)
700 C$=LEFT$(LG$,4)
710 IF C$="REPE" THEN GOTO 930
720 IF C$="ENCO" THEN GOTO 970
730 IF C$="AVAN" THEN GOSUB 1080
740 IF C$="ROTA" THEN GOSUB 1000
750 IF C$="DIST" THEN GOSUB 860:DIST=VAL(RIGHT$(LG$,LEN(LG$)-P))
760 IF C$="DIS+"  THEN GOSUB 1210
770 IF C$="LEVE" THEN LV=1
780 IF C$="BAIS" THEN LV=0
790 PL=PL+1:IF PL>NC THEN END
800 GOTO 690
810 :
820 :
830 :
835 :
840 :
845 :
850 REM ----------- ---------------- recherche espace ------
860 FOR P=1 TO LEN(LG$)
870 IF MID$(LG$,P,1)=" " THEN RETURN
880 NEXT P
890 P=0 : RETURN
900 REM ---------------------------------- repete ----------
910 REM       PR :pt pile RP():contient no ligne apres REPETE
920 REM       NB() contenant nb de boucles
930 GOSUB 860
940 PR=PR+1:RP(PR)=PL+1:NB(PR)=VAL(RIGHT$(LG$,LEN(LG$)-P))
950 PL=PL+1 : GOTO 690
960 REM ----------------- encore --------
970 NB(PR)=NB(PR)-1:IF NB(PR)>0 THEN PL=RP(PR):GOTO 690
980 PR=PR-1:PL=PL+1:GOTO 690
990 REM ---------------------------------- Angle -------------
1000 GOSUB 860
1010 ANG=ANG+VAL(RIGHT$(LG$,LEN(LG$)-P))
1020 IF ANG=>360 THEN ANG=ANG-360
1030 AR=ANG/360*6.28318
1040 CX=COS(AR):CY=SIN(AR)
1045 IF CX>-.01 AND CX<.01 THEN CX=0
1050 RETURN
1060 REM --------------------------------- Trace droite ----
1070 :
1080 GOSUB 860
1090 IF P=0 THEN 1110
1100 DIST=VAL(RIGHT$(LG$,LEN(LG$)-P))
1110 DX=DIST*CX:DY=DIST*CY*1.3
1115 IF LV=1 THEN 1180
1120 CURSET X0,Y0,1
1125 IF DX=0 AND DY=0 THEN PRINT "ERREUR";CD$(PL);:WAIT 300:RETURN
1130 DRAW DX,DY,1
1140 :
1180 X0=X0+DX:Y0=Y0+DY
1190 RETURN
```

```
1200 REM ------------------- Increment distance ---
1210 GOSUB 860
1220 DIST=DIST+VAL(RIGHT$(LG$,LEN(LG$)-P))
1230 RETURN
1240 REM -------------------- Stockage fonction dans f$() ----
1250 NF=NF+1
1260 F$(NF,0)=RIGHT$(X$,LEN(X$)-5)
1270 FOR P=1 TO 10
1280 READ F$(NF,P)
1290 IF F$(NF,P)="FINI" THEN RETURN
1300 NEXT P
1310 STOP
1320 REM ------------------ Insertion fonction ----
1330 P=1
1340 :
1350 IF F$(K,P)="FINI" THEN RETURN
1360 CD$(I)=F$(K,P)
1370 P=P+1:I=I+1:GOTO 1350
```



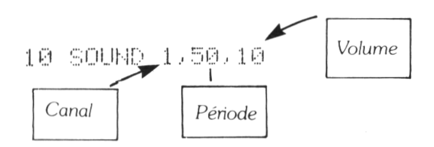
LES SONS

Sur ORIC-1, il existe des sons déjà programmés. Ainsi, en frappant « **EXPLODE** » vous entendez un bruit d'explosion.

Il existe également : **PING**
ZAP
SHOOT

SOUND canal,période,volume :

Cette instruction envoie une fréquence sur le canal spécifie (1, 2 ou 3). La période doit être comprise entre 1 et 32000, le volume entre 0 et 15.



Volume = 0 indique que le volume est contrôlé par PLAY.

Si la commande PLAY n'est pas programmée, seul le canal 1 fonctionne.

```
10 FOR P=2000 TO 1 STEP -50
20 : SOUND 1,P,10
25 : PRINT "FREQUENCE:";(1/P)*100000
27 : WAIT 100
30 NEXT P
```

Pour le bruit, utiliser 4, 5, 6 au lieu de 1, 2, 3.

Music canal,octave,note,volume :

Joue une note sur le canal, spécifie (1, 2 ou 3).

Le niveau d'octave est compris entre 0 et 6, note entre 2 et 12. Pour volume = 0, le volume est contrôlé par PLAY.

Ce programme joue les notes 2 à 12 pour un niveau d'octave.

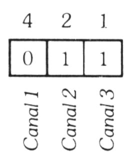
```
5 INPUT "OCTAVE (0-6) ";O
6 :
10 FOR N=2 TO 12
20 :MUSIC 1,O,N,10
25 :WAIT 50
30 NEXT N
40 :
60 GOTO 5
```

Play voies autorisées,voies bruit,type enveloppe, période enveloppe :

Play contrôle la correction des canaux, le type d'enveloppe et la période de celle-ci.

Voies autorisées :

Le codage interne des voies est fait en binaire sur 3 bruits, chaque bruit représente une voie.



La commande des canaux 1 et 2 se fait par :

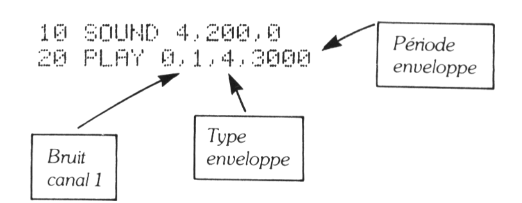
3 = 2 + 1 → (PLAY 3,...)
Pour 1, 2 et 3 7 = 4 + 2 + 1 (PLAY 7,...)

Voies bruit :

Le principe de codage des voies est le même que ci-dessus.

Exemple :

Bruit de vagues sur le canal 1.



Type Enveloppe :
Période Enveloppe :

Lorsque le volume spécifie dans SOUND ou MUSIC est égal à ZERO, la forme et la période du signal sont déterminées par « Type enveloppe » et « période enveloppe ».

1
2
3
4
5
6
7

DUREE FINIE

CONTINU

```
10 PLAY 1,0,4,2000
20 SOUND 1,100,0
```

```
10 REM ESSAI ENVELOPPE
20 :
30 INPUT "ENVELLOPPE ";E
40 INPUT "PERIODE ENVELOPPE ";P
50 :
60 MUSIC 1,3,4,0      :REM vol=0 -> controle par PLAY
70 :
80 PLAY 1,0,E,P
90 GOTO 30
```

```
10 PLAY PLAY 0,0,0,0
```
Stoppe tous les sons

Morse

Ce programme vous permet d'apprendre l'alphabet MORSE.
Par exemple, en appuyant sur la lettre S, vous entendez 3 sons brefs.
En appuyant sur 0, vous entendez 3 sons longs

```
10 REM       MORSE   25.3.83
12 :
15 PRINT CHR$(6);          :REM Clavier muet
20 :
30 DATA .-,-...,-.-.,-..,.,..-.
40 DATA --.,....,..,.---,-.-,.-..,--
50 DATA -.,---,.--.,--.-,.-.,...,-,..-
60 DATA ...-,.--,-..-,-.--,--..
70 DIM MRS$(26)
80 FOR P=1 TO 26
90 : READ MRS$(P)
100 NEXT P
110 :
120 GET X$
130 :
140 P=ASC(X$)-64:IF P=3-64 THEN PRINT CHR$(6):END
145 :
150 IF P<1 OR P>26 THEN 140
160 X$=MRS$(P)
165 :
170 FOR P=1 TO LEN(X$)
180 : IF MID$(X$,P,1)="." THEN SOUND 1,500,13:WAIT 5:SOUND 1,500,0
190 : IF MID$(X$,P,1)="-" THEN SOUND 1,500,13:WAIT 30:SOUND 1,500,0

200 : WAIT 10
210 NEXT P
220 GOTO 120
230 REM-----------------------------------------
240 REM Pour afficher en MORSE
250 REM faire: 165 PRINT MRS$(P);"  ";
```

A	• —
B	— •••
C	— • — •
Z	— — ••

Table MRS$ (   )

Pian'Oric

Ce programme permet de jouer des notes à partir du clavier. Le programme s'arrête lorsque vous appuyez sur « CTRLC » (code 3).

```
10 CLS
15 PRINT CHR$(6)            :REM CLAVIER MUET
20 DIM CL$(40),O(40),N(40)
30 :
40 DATA Q,3,2
50 DATA W,3,3
60 DATA E,3,4
62 DATA R,3,5
64 DATA T,3,6
70 :
100 DATA FIN
110 :
200 FOR P=1 TO 40
210 : READ X$:IF X$="FIN" THEN 300
220 : READ OT,NT
230 : CL$(P)=X$:O(P)=OT:N(P)=NT
240 :NEXT P
250 :
300 X$=KEY$:IF X$="" THEN 300
305 IF ASC(X$)=3 THEN PRINT CHR$(6):END
310 :
320 FOR I=1 TO P
330 :IF CL$(I)<>X$ THEN 360
335 :MUSIC 1,O(I),N(I),10
337 :PLAY 1,0,0,0
340 :WAIT 30
350 :PLAY 0,0,0,0
360 NEXT I
370 GOTO 300
```

CL$	*O( )*	*W( )*
Q	3	2
W	3	3
E	3	4
R	3	5

ANNEXE 1
RÉCAPITULATIF DES INSTRUCTIONS BASIC

	ABS (– 5)	Donne la valeur absolue d'un nombre PRINT ABS (– 5) → 5
	AND	Permet le test de plusieurs conditions: Les instructions après THEN sont exécutées. SI condition 1 ET condition 2 sont VRAIES: 90 INPUT "X ?"; X 100 IF (X > 0) AND (X < 10) THEN PRINT "X est supérieur à 0 et inférieur à 10"
	ASC ("B") ASC ("CHARLIE")	Donne le code ASCII d'un caractère ou du premier caractère d'une chaîne (qui ne doit pas être vide) PRINT ASC ("B") → 66 PRINT ASC ("CHARLIE") → 67
	ATN (X)	Fournit l'arc tangeante de X exprimé en radians.
	CALL X	Appelle le sous-programme d'adresse X. Retourne en séquence après un RTS.
	CHAR code, jeu, coul.	En haute résolution, dessine le caractère de code spécifique, à la position courante du curseur. 'jeu' (0 ou 1) spécifie le jeu standard ou alterne. 'code' doit être compris entre 32 et 127. CHAR 65,0,1 dessine le caractère A.
	CHR$(66) CHR$(7)	Donne un caractère dont le code est 66. C'est-à-dire B PRINT CHR$(66) Affiche le caractère B PRINT CHR$(7); Active la sonnerie du clavier.
	CIRCLE rayon, coul.	Dessine un cercle centre sur la position du curseur. 0: BACKGROUND 1: FOREGROUND 2: INVERSE
	CLEAR	Initialise les variables à 0.
	CLOAD "PROG" CLOAD "PROG",S	Charge le programme PROG en vitesse rapide. Charge "PROG" en vitesse lente.

	CLS	Efface l'écran.
	CONT	Réactive l'exécution d'un programme interrompu par 'BREAK' ou 'Ctrl C' ou STOP dans un programme.
	CURMOV Dx,Dy, coul.	Déplace le curseur sans dessiner.
	CSAVE "PROG" CSAVE "PROG",S	Sauve le programme PROG sur cassette en vitesse rapide. Sauve PROG sur cassette en vitesse lente.
	CURSET X,Y,coul.	Positionne le curseur au point X,Y X doit être compris entre 0 et 239 Y doit être compris entre 0 et 199
	DATA "Janvier:",83	Définit des données qui sont lues par READ 10 DATA "Janvier:",83 20 READ MOIS$:READ AN 30 PRINT MOIS$,AN → Janvier 83
	DEEK(adresse)	Fournit en décimal le contenu de 2 octets successifs.
	DEF FNx(X)=...	Définit une fonction de X. 10 DEF FNA(X)=X/2 20 PRINT FNA(12) → 6
	DIM MOIS$(12) DIM MOIS(12) DIM VNTE(12,5)	Réserve la place pour une table MOIS$() (Une table non dimensionnée par DIM est dimensionnée à 10 par BASIC)
	DOKE adresse,valeur	Range une valeur décimale dans 2 octets consécutifs.
	DRAW DX,DY,type coul.	Dessine une droite à partir du point courant 0: BACKGROUND 1: FOREGROUND 2:INVERSE
	ELSE	IF cond THEN instruction ELSE instruction Exécute la ou les instructions après THEN si la condition testée est VRAIE. Exécute la ou les instructions après ELSE si la condition testée est FAUSSE.
	END	Définit la fin du programme
	EXP(X)	Donne l'exponentielle de X
	EXPLODE	Provoque un bruit d'explosion.
	FALSE	Donne 0.
	FILL nlignes,ncol,coul	Remplit 'nlignes' de l'écran avec la couleur spécifiée à droite du curseur. L'écran comporte 200 lignes de 40 colonnes.

	FOR I=1 TO 5 : NEXT I	Toutes les instructions entre FOR et NEXT sont exécutées 5 fois. 10 FOR I=1 TO 5 20 PRINT I,I*I 30 NEXT I 1 1 2 4 . . 5 25
	FOR I=3 TO 1 STEP−1 : NEXT I	10 FOR I=3 TO I STEP −1 20 PRINT I*I,I 30 NEXT I 3 9 2 4 1 1
	FRE(0)	Donne la place libre en mémoire centrale. PRINT FRE(0) → 15234
	GET X$	Lit un caractère au clavier (sans l'afficher) 10 GET X$ 20 PRINT X$; 30 GOTO 10
	GOSUB 1000	Provoque un branchement à l'instruction 1000 (comme GOTO 1000) mais dès qu'une instruction RETURN est rencontrée, l'exécution se poursuit en 110. 100 GOSUB 1000 → 1000 sous prog 110 .... 1010 ... 200 GOSUB 1000 1020 ... 210 .... 1030 RETURN
	GOTO 1000	Provoque un branchement ou programme en 1000. Peut être utilisé en 'mode immédiat' après une interruption : les variables ne sont pas remises à zéro (comme le ferait RUN 1000)
	HEX$(valeur)	Affiche une valeur décimale en hexadécimal. PRINT HEX$(20) → 14
	HIRES	Provoque le passage en haute résolution. Le curseur est positionné en 0,0.
	IF cond THEN instruct	Teste une condition et exécute la ou les instructions après THEN si la condition est VRAIE. 10 INPUT "Votre âge ?" ;AGE 20 IF (AGE>18) THEN PRINT "Vous êtes majeur"

	INPUT "Année?";AN INPUT "Nom?";NOM$ INPUT "Nom,Age?";NOM$,AN	Affiche le message 'Année?' puis attend que l'opérateur entre la valeur de AN. Affiche le message 'Nom,Age?' puis attend que l'opérateur entre les valeurs de NOM$ et de AGE (séparés par une virgule). Une chaîne ne doit pas comporter de virgule.
	INT(valeur)	Donne la partie entière d'une valeur. PRINT INT(12.6) → 12
	KEY$	Lit le clavier en permanence et retourne une chaîne vide si aucun caractère n'a été frappé. 10 X$=KEY$:IF X$="" THEN 10 20 PRINT X$;ASC(X$) 30 GOTO 10
	LEFT$("BASIC",3)	Donne les caractères de gauche d'une chaîne PRINT LEFT$("BASIC",3) → BAS
	LEN("BASIC")	Donne la longueur d'une chaîne de caractères. PRINT LEN("BASIC") → 5
	LIST LIST 10 LIST 10-30	Liste tout le programme Liste la ligne 10 Liste les lignes 10 à 30
	LN(X) LOG(X)	Donne le logarithme naturel. Donne le logarithme à base 10.
	LORES 0 LORES 1	Passagé en basse résolution jeu de caractères normal. Jeu de caractères alterné (semi-graphiques).
	LPRINT "Bonjour"	Affiche 'Bonjour' sur l'imprimante.
	MID$(X$,7,4)	Donne les caractères du milieu d'une chaîne. 10 X$="BASIC POUR TOUS" 20 PRINT MID$(X$,7,4) → POUR
	MUSIC canal,octave,note,vol.	Joue la note spécifiée.
	NEW	Efface le programme résidant en mémoire centrale.
	ON B GOSUB 100,200,500	Suivant la valeur de B (1,2,3), le programme s'exécute en 100, 200 ou 500. Au retour du sous-programme, le programme continue à s'exécuter après l'instruction ON B GOSUB......... 10 INPUT "Votre choix? (1,2,3)?";CH 20 ON CH GOSUB 100,200,500 30 END 100 PRINT "Choix 1":RETURN 110 : 200 PRINT "Choix 2":RETURN 210 : 500 PRINT "Choix 3":RETURN

	ON B GOTO 100,200,500	Même chose que pour ON B GOSUB... Mais sans retour.
	OR	Permet le test de plusieurs conditions. Si cond1 OU cond2 est VRAIE, Les instructions après THEN sont exécutées. 90 INPUT "X?";X 100 IF (X=1) OR (X=3) THEN PRINT "X est "égal à 1 OU 3"
	PAPER no couleur	Détermine la couleur de fond de l'écran. 'no couleur' est compris entre 0 et 7.
	PATTERN nombre	Permet de tracer des pointillés dont l'image correspond à la valeur du nombre en binaire. 'nombre' doit être compris entre 0 et 255.
	PEEK(1000)	Fournit le contenu de la mémoire d'adresse 1000. 10 X=PEEK(1000)
	PI	PRINT PI → 3.14159265
	PING	Produit un son.
	PLOT X,Y,17 PLOT X,Y,"AAA"	En basse résolution, affiche un carré rouge en X,Y. Affiche 'AAA' en X,Y.
	POINT(X,Y)	Donne −1 si le point X,Y est allumé, 0 s'il est éteint.
	POKE 1000,12	Range la valeur 12 (décimal) dans la mémoire d'adresse 1000.
	PRINT PRINT "BONJOUR" PRINT "DUPONT";TAB(9);"JEAN"	Provoque un saut de ligne. Affiche BONJOUR Affiche DUPONT puis JEAN en colonne 9
	READ MOIS$,AN	cf DATA
	REM blabla	Tout ce qui suit REM est du commentaire.
	REPEAT	Les instructions entre REPEAT et UNTIL sont exécutées jusqu'à ce que la condition testée par UNTIL soit vérifiée. 10 REPEAT 20 : PRINT "Appuyez sur une touche" 30 UNTIL KEY$<>"" 40 50 PRINT "Boucle terminée"
	RESTORE	Positionne le pointeur des DATAS au début. Permet ainsi leur relecture.
	RETURN	Signale la fin d'un sous-programme. Provoque un retour au programme appelant (à l'instruction qui suit GOSUB)

	RIGHT$(X$,3)	Donne les caractères de droite d'une chaîne. 10 X$="BASIC" 20 PRINT RIGHT$(X$,3) → SIC
	RND(1) RND(−3) RND(0)	Fournit un nb aléatoire entre 0 et 1 Initialise une série aléatoire Fournit le dernier nombre aléatoire.
	RUN RUN 100	Provoque l'exécution du programme. Initialise les valeurs des variables à zéro. Provoque l'exécution à partir de la ligne 100.
	SCRN (X,Y)	Donne le code ASC II au point X, Y
	SGN(X)	Donne le signe d'un nombre : 1 positif 0 nul −1 négatif
	SHOOT	Produit un son.
	SIN(X)	Donne le sinus de X exprimé en radians.
	SOUND canal,période,vol	Produit un son avec la période spécifiée.
	SPC(3)	Imprime 3 espaces. 10 PRINT "XXX";SPC(3);"XXX" → XXX XXX
	SQR(X)	Donne la racine carrée de X qui doit être positif
	STOP	Stoppe l'exécution qui peut être poursuivie en frappant CONT.
	STR$(123)	Convertit un nombre en chaîne : 10 X=123:X$=STR$(X) 20 PRINT RIGHT$(X$,1) → 3
	TAB(15)	Positionne le curseur en colonne 15 de la ligne courante. PRINT "LIBERT";TAB(15);"Pierre" → LIBERT Pierre
	TEXT	Annule le mode graphique.
	TROFF	Annule la trace (TRON)
	TRON	Permet de visualisers les n^os des instructions exécutées (est annulé par TROFF).
	VAL(X$)	Donne la valeur numérique d'une chaîne : 10 X$="123 FRANCS" 20 PRINT VAL(X$) → 123
	WAIT temps en 1/100e sec.	Provoque une pause du programme.
	ZAP	Produit un son.

ANNEXE 2
MODIFICATION DE PROGRAMME

La modification de programme se fait en se positionnant d'abord sur la ligne puis en recopiant la ligne à l'aide de <CTRL A>.

```
CLS
LIST

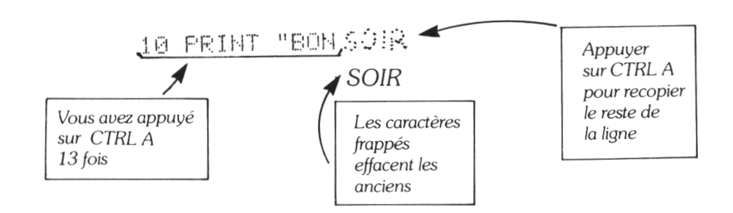
10 PRINT "BONJOUR"
20 PRINT "MONSIEUR"
```

Positionnement du curseur sur la ligne avec ↑

Vous appuyez autant de fois qu'il le faut pour vous positionner sur la ligne à modifier.

Recopie des caractères de la ligne avec CTRL A :

Lorsque vous êtes positionné sur la ligne vous recopiez la partie de la ligne inchangée en appuyant sur <CTRL A>.



Remplacement des caractères :

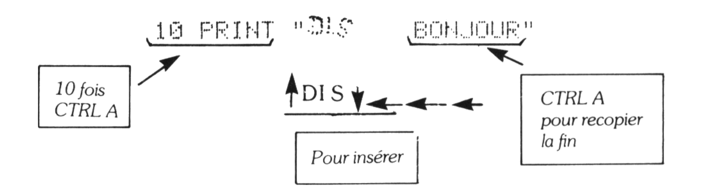
Il suffit d'écrire les nouveaux caractères sur les anciens.

Suppression de caractères : →

En frappant '→' au lieu de CTRL A, on 'saute' les caractères qui ne doivent pas être recopiés.

Insertion de caractères :

Pour insérer des caractères dans une ligne (DIS sur l'exemple) il faut frapper ↑ DIS ↓ ←←←



On remarque qu'une ligne peut être recopiée avec un autre numéro.

EDIT n° ligne:

affiche la ligne spécifiée.

ANNEXE 3
MESSAGES D'ERREURS

Si le message envoyé par Basic ne suffit pas pour repérer l'erreur, pensez à **visualiser en mode immédiat** les valeurs des variables (avant de modifier le programme puisqu'une modification initialise les variables à zéro).

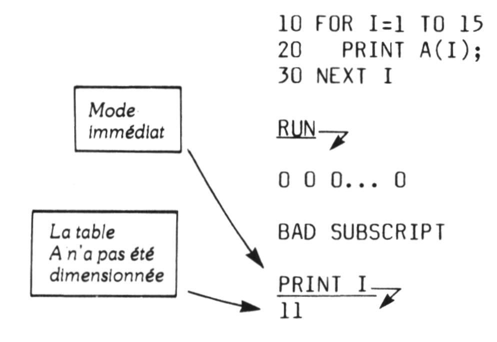
BAD SUBSCRIPT

On cherche à référencer un élément en dehors d'une table.

```
10 DIM A(13)
   :
70 PRINT A(15)
```

Il faut dimensionner la table avec 15 éléments.

Cette erreur se produit également lorsque la table n'a pas été dimensionnée par DIM (elle est alors dimensionnée à 10 par BASIC). Dans ce cas, PRINT A(11) provoque une erreur.



CAN'T CONTINUE (ne peut continuer)

L'exécution du programme ne peut être poursuivie :
— Une instruction a été modifiée.

DIVISION BY ZERO (division par zéro)

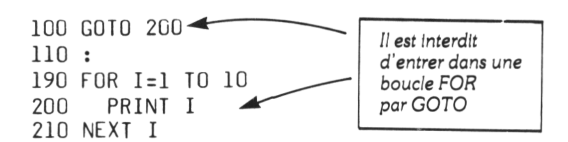
Division par zéro impossible (ajouter un test IF X = 0 THEN...)

ILLEGAL DIRECT

Cette instruction ne peut être utilisée en mode direct (INPUT par exemple).

NEXT WHITOUT FOR (NEXT sans FOR)

Une instruction NEXT est rencontrée sans qu'une instruction FOR ait été exécutée auparavant.



Il faut faire : 100 GOTO 190

OUT OF DATA (plus de DATAS)

On essaie de lire des DATAS avec READ alors que toutes les DATAS ont déjà été lues.

OUT OF MEMORY (plus de place mémoire)

OVERFLOW (débordement)

La valeur d'un nombre dépasse la capacité prévue.

REDIMENSIONNED ARRAY (redimensionnement de table)

On essaie de dimensionner une table qui l'a déjà été.
On a déjà référencé un élément d'une table, ce qui a provoqué son dimensionnement à 10 par Basic, puis on la dimensionne explicitement.

```
10 A(5)=123
20 DIM A(15)

RUN

REDIMENSIONNED ARRAY
```

RETURN WHITOUT GOSUB (retour sans appel)

Une instruction RETURN a été exécutée sans qu'il y ait eu une instruction GOSUB exécutée auparavant.

```
 10 GOSUB 100
 30 :
100 PRINT "sous-programme"
110 RETURN
```

On a oublié END ou STOP après 10 : le sous-programme est exécuté 2 fois. (Il faut ajouter : 20 STOP).

STRING TOO LONG (chaîne trop longue)

Une chaîne devient supérieure à 255 caractères.

```
10 X$=X$+"A"
20 GOTO 10
```

SYNTAX ERROR (erreur de syntaxe)

TYPE MISMATCH

On essaie d'attribuer une valeur numérique à une chaîne.

10 X$ = 123 (il faut faire X$ = « 123 » ou X = 123)

10 X = « AZERTY » (il faut faire X$ = « AZERTY »)

UNDEF'D STATEMENT (no de ligne indéfini)

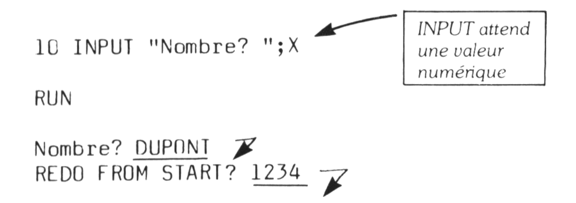
Une instruction GOTO ou GOSUB référence une ligne qui n'existe pas.

UNDEF'D FUNCTION (fonction indéfinie)

On appelle une fonction qui n'a pas été définie.

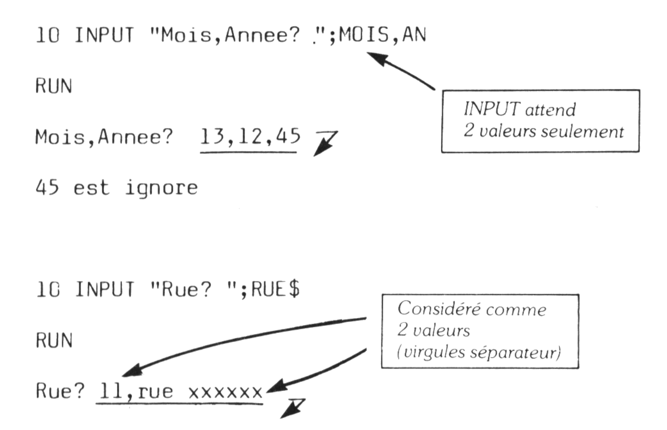
REDO FROM START (recommencer depuis le début)

On essaie d'entrer une chaîne dans une instruction INPUT alors que celle-ci attend une valeur numérique.



EXTRA IGNORED

On essaie d'introduire une donnée de plus que prévu.



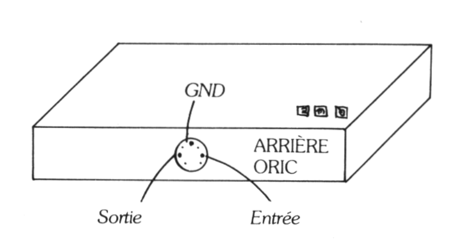
Seul 11 est accepté (ne pas entrer la virgule)

ANNEXE 4
SAUVEGARDE DES PROGRAMMES SUR CASSETTE :

La prise magnéto est à 7 broches :
— 3 pour l'enregistrement/lecture
— 2 pour la commande moteur du magnéto
— 2 pour la HI-FI

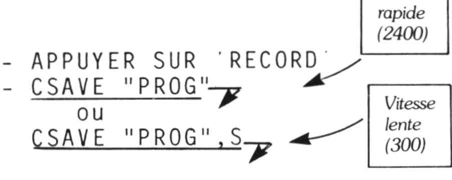
Seules les 3 premières broches sont indispensables. Mais il faut alors commander soi-même le magnétophone.

Avec le magnétophone utilisé (Thomson), nous n'avons pas eu de problème, le réglage du niveau n'est pas critique. Le cordon doit être court et blindé





Sauvegarde :



La lampe témoin du magnétophone doit s'allumer.

Lecture :

```
- CLOAD "PROG"
     ou
  CLOAD "PROG",S
     ou
  CLOAD "",S
- APPUYER SUR 'PLAY'
```

ANNEXE 5
CARACTÈRES DE CONTRÔLE :

Clavier:

<**CTRL T**> Clavier en mode majuscule (CAPS) ou minuscules (BASCULE)
<**CTRL P**> Imprimante
<**CTRL F**> Clavier muet ou sonore (BASCULE) - CHR$ (6)
<**CTRL D**> Double hauteur
<**CTRL Q**> Curseur présent ou absent (BASCULE)

Ecran:

<**CTRL J**> Saut de ligne (CHR$ (10))
<**CTRL L**> Effacement écran
<**CTRL M**> Retour chariot

ANNEXE 6
CARACTÈRES SPÉCIAUX

Exemple : PRINT " "; CHR$ (27); «B»; «VERT»

CHR$ (27) +

@	FOREGROUND NOIR
A	ROUGE
B	VERT
C	JAUNE
D	BLEU
E	VIOLET
F	BLEU CLAIR
G	BLANC
H	RETOUR EN MODE NORMAL
I	ACCES AUX CARACTERES ALTERNES
J	HAUTEUR DOUBLE FIXE
K	
L	HAUTEUR NORMALE CLIGNOTANT
M	
N	HAUTEUR DOUBLE CLIGNOTANT
O	
P	BACKGROUND NOIR
Q	ROUGE
R	VERT
S	JAUNE
T	BLEU
U	VIOLET
V	BLEU CLAIR
W	BLANC

ANNEXE 7
TABLE DES CODES ASCII

0	ec	32	SPACE	64	e	96	`
1	Ac	33	!	65	A	97	a
2	Bc	34	``	66	B	98	b
3	Cc	35	≠	67	C	99	c
4	Dc	36	$	68	D	100	d
5	Ec	37	%	69	E	101	e
6	Fc	38	&	70	F	102	f
7	Gc (BEL)	39	`	71	G	103	g
8	Hc (BS)	40	(	72	H	104	h
9	Ic (HT)	41	)	73	I	105	i
10	Jc (LF)	42	*	74	J	106	j
11	Kc	43	+	75	K	107	k
12	Lc	44	`	76	L	108	l
13	Mc (CR)	45	–	77	M	109	m
14	Nc	46	.	78	N	110	n
15	Oc	47	/	79	O	111	o
16	Pc	48	0	80	P	112	p
17	Qc	49	1	81	Q	113	q
18	Rc	50	2	82	R	114	r
19	Sc	51	3	83	S	115	s
20	Tc	52	4	84	T	116	t
21	Uc	53	5	85	U	117	u
22	Vc	54	6	86	V	118	v
23	Wc	55	7	87	W	119	w
24	Xc	56	8	88	X	120	x
25	Yc	57	9	89	Y	121	y
26	Zc	58	:	90	Z	122	z
27	[c (ESC)	59	;	91	[	123	{
28	/c	60	<	92	/	124	:
29	]c	61	=	93	]	125	}
30	ˆc	62	>	94	ˆ	126	-
31	-c	63	?	95	—-	127	DEL

Achevé d'imprimer en janvier 1984
sur les presses de l'imprimerie Laballery et Cie
58500 Clamecy
Dépôt légal : janvier 1984
No d'impression : 312030
No d'édition : 86595-98-2
ISBN : 2-86595-098-0

ORIC-1 pour tous

Apprenez l'ORIC-1 — L'"ORIC-1 pour tous" en main, placez-vous devant votre machine et commencez à écrire quelques instructions.
Très vite, vous assimilerez les notions fondamentales de la programmation : variables, tests, boucles...
Vous pourrez alors aborder la "magie" de l'ORIC, les **graphiques** et les **sons**, grâce aux nombreux exemples illustrés et aux programmes directement commentés.
Sur ces bases, il vous sera aisé d'approfondir vos connaissances et d'écrire vos propres programmes (gestion, enseignement, jeux...).



EDITIONS DU P.S.I.
B.P. 86 - 77402 LAGNY/MARNE CEDEX
FRANCE

***

Couverture : Henri Moulierac


ORIC-1 pour tous
JACQUES BOISGONTIER
SOPHIE BREBION